République Algérienne Démocratique et Populaire.
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

Université Saad Dahlab, Blida
USDB.

Faculté Des Sciences.
Département Informatique.

Mémoire Pour L'Obtention
D'un Diplôme D'Ingénieur D'Etat en Informatique.
Option : Système d'information

Sujet :

Conception et réalisation d'un système de gestion d'un produit X25 par les EJB

Présenté par : ZOUGGAGH Mohamed **- Promoteur : Mr.BENNOUAR DJAMEL**

Soutenue le: 15/10/2005, devant le jury composé de :

Président :

Examinateur :

Examinateur :

Promotion N° 03 / 2004 - 2005

Remerciements

Je remercie avant tout le bon Dieu qui m'a aidé à réaliser ce modeste travail.

Je tien à remercier Mr Bennouar pour m' avoir encadré et pour son aide.

Je remercie Meme le chef du département d'Informatique, tous les enseignants de la faculté des sciences de BLIDA et surtout ceux du département informatique.

Je remercie les membres du jury pour m'avoir donné l'honneur de juger mon travail.

Nous remercions, de tout coeur, tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Z. mohamed

Dédicaces

Je dédie ce travail :

A mes parents qui m'ont soutenu durant toutes mes études, qui ont été à mes cotes dans le meilleur et dans le pire, et ont été toujours pour moi une source d'affection et d'amour

A mes frères Redha, Hamza, Nassim, mes oncles, mes tantes, et mes cousins.

A toute la promotion d'informatique 2004 / 2005 en général et a Ali, Amin, Talab, Mohamed, Ahmed , Brahim et Amel en particulier

Z.Mohamed

Résumer

Le but de cette mémoire est l'application d'**Entreprise Java Bean (EJB)** sur un système de gestion commercial pour un produit X25. Noter que les EJB ne seront pas appliqués sur tout le système mais sur une partie qui sera définie dans le chapitre étude du système.

The goal of this memory is the application of **Enterprise Java Bean (EJB)** on a commercial system of management for a product X25. To note that the EJB will not be applied to all the system but to a part which will be defined in the chapter study of the system.

TABLE DES MATIERES

LISTE DES FIGURES

Introduction générale

I. Historique

Au début des années 1970, les premiers grands systèmes informatiques se composent d'ordinateurs centraux (*mainframe*) volumineux et fragiles, auxquels accèdent en temps partagé des terminaux. Un terminal est composé d'un écran et d'un clavier mais sans puissance de calcul. Ces systèmes constituent en quelque sorte les premiers réseaux informatiques. Les applications qui tournent sur ces systèmes sont des applications « *un tiers* » du fait que les trois couches d'abstraction d'une application sont intimement liées et s'exécutent sur le même ordinateur.

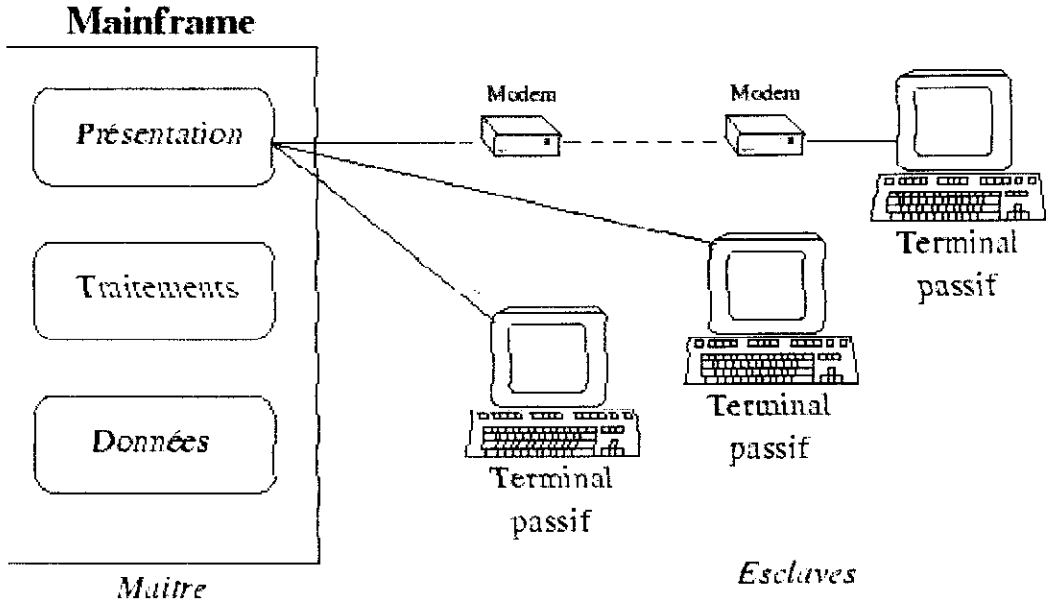


Figure 1.1 : Architecture sur site central [VAN99]

Au cours des années 1980, et avec l'arrivée en masse des micro-ordinateurs qui ont une puissance de calcul, le monde informatique a complètement changé. Les grands systèmes sont alors massivement décentralisés, si bien que l'importance des réseaux informatiques s'en trouvent multipliés, de par le nombre de machines connectées, les quantités de données échangées et la diversité de nature des communications.
La plupart des réseaux informatiques sont construits selon une architecture client / serveur « Architecture deux tiers », où chaque machine peut confier des tâches aux autres ordinateurs en dialoguant, *via le* réseau, suivant des protocoles standardisés.

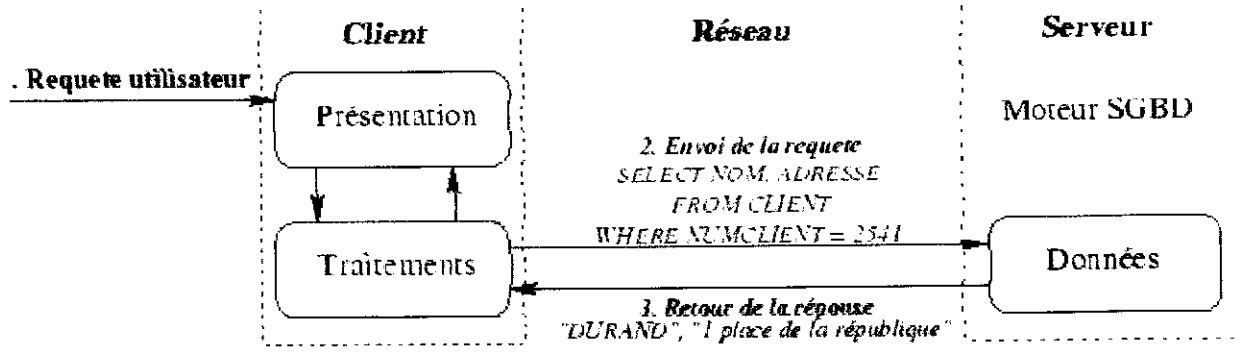


Figure 1.2 : Architecture client / serveur [VAN99]

L'expérience a démontré qu'il était coûteux et contraignant de vouloir faire porter l'ensemble des traitements applicatifs par le poste client. On arrive aujourd'hui à ce que l'on appelle le client lourd, ou *fat client*. [VAN99]
Malgré tout, l'architecture deux tiers présente de nombreux avantages qui lui permettent de présenter un bilan globalement positif :

- *Elle a permis l'utilisation d'une interface utilisateur riche*,
- elle a permis l'appropriation des applications par l'utilisateur,
- elle a introduit la notion d'interopérabilité. [VAN99]

Pour résoudre les limitations du client serveur tout en conservant ses avantages, La réponse est apportée par les architectures trois tiers. Dans ce cas, l'architecture trois tiers applique les principes suivants :

- les données sont toujours gérées de façons centralisées,
- la présentation est toujours prise en charge par le poste client,
- la logique applicative est prise en charge par un serveur intermédiaire (***Serveur d'applications***). [VAN99]

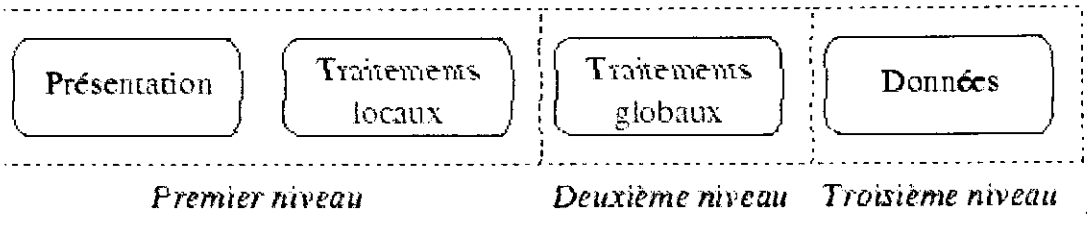


Figure 1.3 : Architecture trois tiers [VAN99]

II. Serveurs d'applications

Les serveurs d'applications sont aujourd'hui l'un des composants logiciels les plus importants.
SUN a proposé une architecture *standard* de ce que se doit d'être un serveur d'application J2EE (Java 2 Entreprise Edition):
Un serveur d'application selon la spécification J2EE est un ensemble de services et d'API permettant l'hébergement d'applications d'entreprise.
Le serveur d'application prend en charge l'ensemble des fonctionnalités qui permettent à N clients d'utiliser une même application

- *Gestion de la session utilisateur* : N clients utilisant une même instance d'application sur le serveur, il est nécessaire que le serveur d'application puisse conserver des contextes propres à chaque utilisateur.
- *Gestion des montées en charge et reprise sur incident* : Afin de gérer toujours plus d'utilisateurs, le serveur d'application doit pouvoir se déployer sur plusieurs machines et éventuellement offrir des possibilités de reprise sur incident.
- *Ouverture sur de multiples sources de données* : C'est le serveur d'application qui rend accessible les données des applications du système d'information. Il doit donc pouvoir accéder à de nombreuses sources de données.

Le schéma suivant reprend les principaux composants de cette architecture:

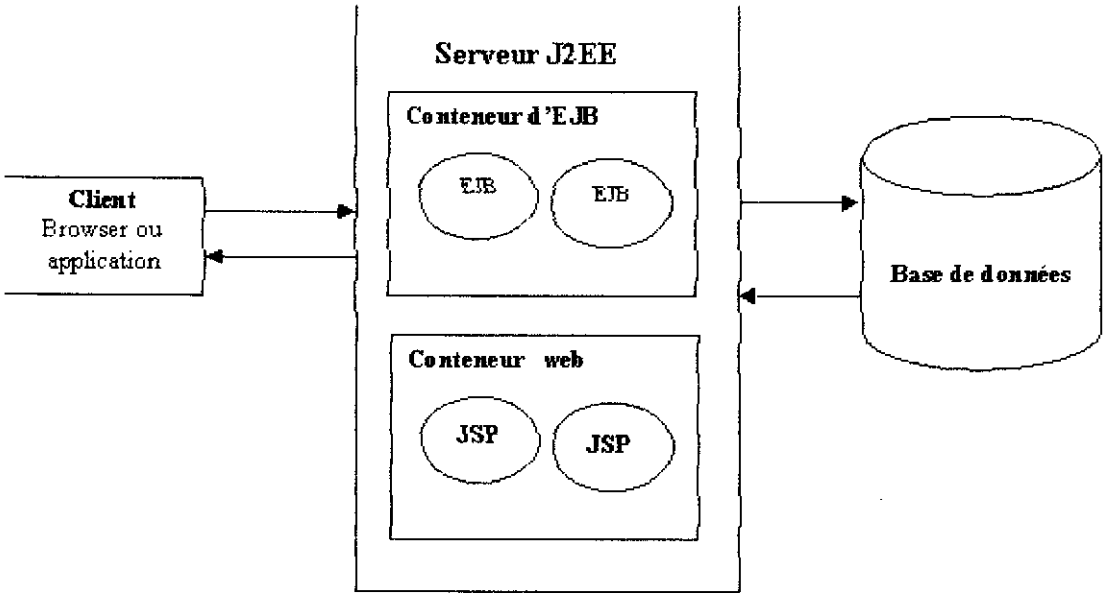


Figure 1.4 : Les composants de l'architecture j2EE

Les grands éditeurs de serveurs d'applications à l'exception de Microsoft ont suivi ces spécifications :

Figure 1.5 : Les Editeur de serveur d'application certifié J2EE

La communauté **Open Source**, comme à son habitude, a développé une multitude de projets intégrant différentes solutions en un environnement de développement J2EE :

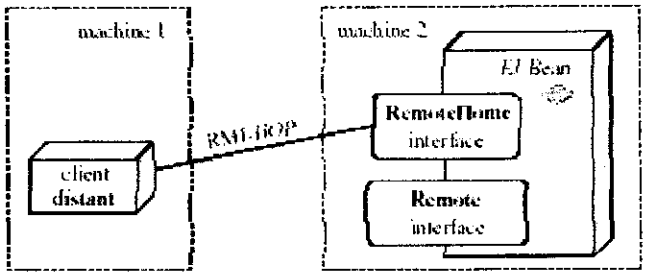
- Ant (projet Jakarta) comme fédérateur.
- JUnit pour les testes unitaires.
- Easy Struts (ou Struts console pour la partie JSP/Servlet).
- XDoclets pour la génération du code et des descripteurs J2EE
- MiddleGen pour les EJB et le mapping vers le SGBD-R
- Cactus (projet Jakarta) pour les testes des composants serveurs
- JMeter (projet Jakarta) pour les testes de montée en charge

CHAPITRE I :

Entreprise Java Bean

I. Définition des EJB :

Les EJB sont en fait des objets distants utilisant RMI/IIOP (un mélange de la syntaxe RMI et du protocole de communication utilisé par Corba) : cela permet de garantir que les EJB pourront être utilisés via un quelconque code client.



Ces composants permettent de mettre en oeuvre la logique métier d'une application codée suivant une architecture logicielle 3 tiers ou plus.

II. Topologie des composants EJB :

Il existe deux types d'EJB : les beans de session (session beans) et les beans entité (les entity beans). Depuis la version 2.0 des EJB, il existe un troisième type de bean : les beans orienté message (message driven beans).
Ces trois types de bean possèdent des points communs notamment celui de devoir être déployés dans un conteneur d'EJB.
Les session beans peuvent être de deux types : sans état (stateless) ou avec état (statefull).
Les beans de session sans état peuvent être utilisés pour traiter les requêtes de plusieurs clients. Les beans de session avec état ne sont accessibles que lors d'un ou plusieurs échanges avec le même client. Ce type de bean peut conserver des données entre les échanges avec le client.
Les beans entité assurent la persistance des données. Il existe deux types d'entity bean :

- persistance gérée par le conteneur (CMP : Container Managed Persistence)
- persistance gérée par le bean (BMP : Bean Managed Persistence).

Avec un bean entité CMP (container-managed persistence), c'est le conteneur d'EJB qui assure la persistance des données. Un bean entité BMP (bean-managed persistence), assure lui même la persistance des données grâce à du code inclus dans le bean.
La spécification 2.0 des EJB définit un troisième type d'EJB : les beans orientés message (message-driven beans). [03]

III. Cycle de vie :

a. Session bean (statefull):

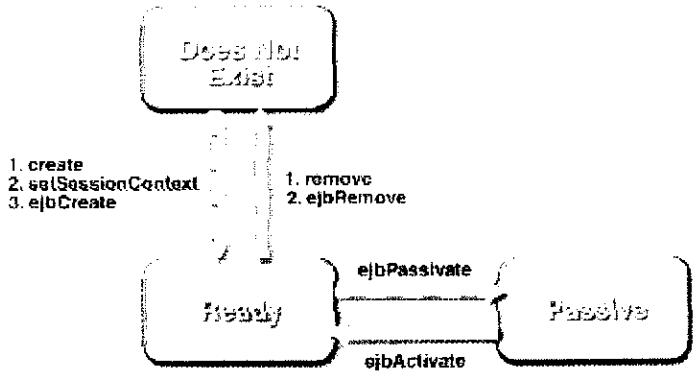


Figure 2.1 : cycle de vie de statfull bean [TJT04]

Les méthodes **create** et **remove** sont invoquées par le client.
Le conteneur sauvegarde l'état du bean lors du passage de **Ready** ver **Passive**

b. session bean (Stateless):

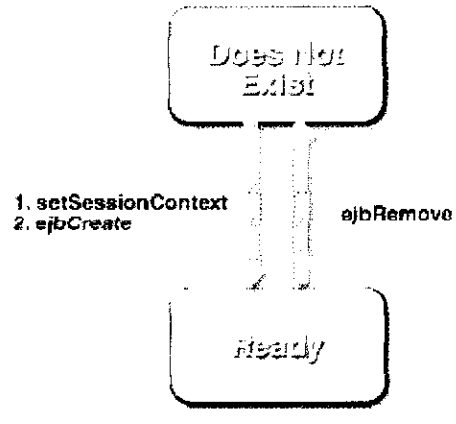


Figure 2.2 : cycle de vie de stateless bean[TJT04]

Le conteneur ne sauvegarde pas l'état du bean

c. Entity bean :

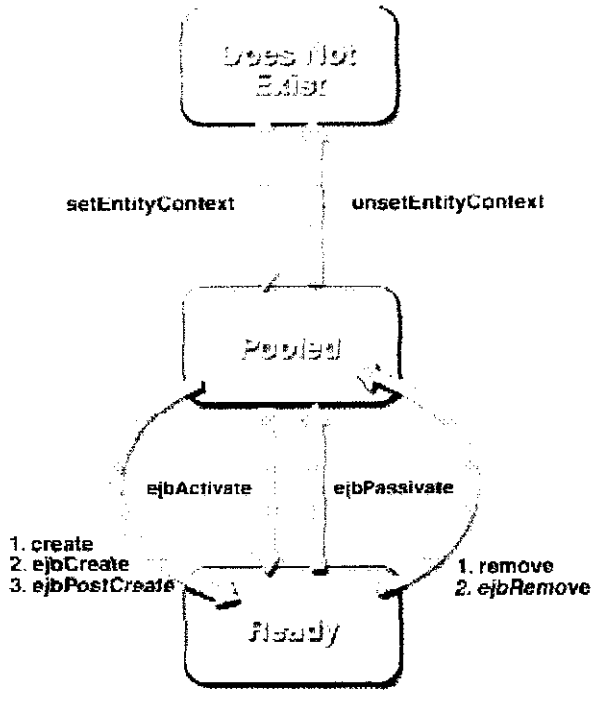


Figure 2.3 : cycle de vie de Entity bean [TJT04]

IV. Environnement EJB :

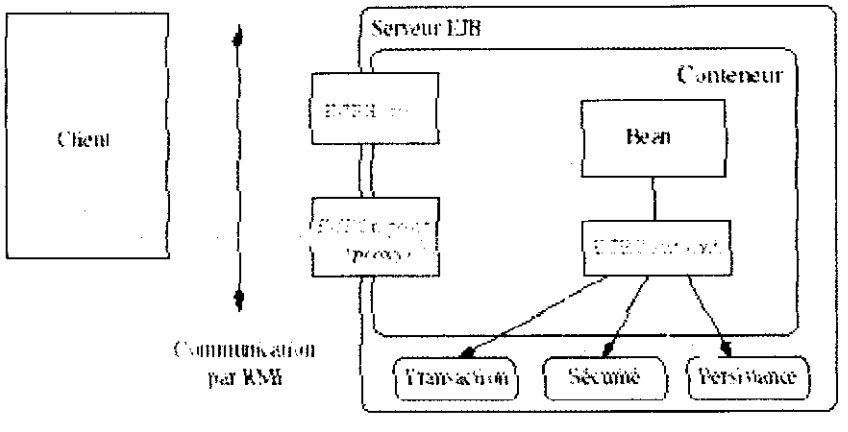


Figure 2.4 : *Un appel de méthode d'un composant EJB* [CCC03]

La figure 2.4 donne une vue d'un appel de méthode d'un composant EJB de type Session ou Entité.

On retrouve la fabrique de composant et la classe de délégation. La première est une classe qui dérive de EJBHome, plusieurs méthodes de type **create** et **remove** permettent de créer ou supprimer une instance. La délégation est assurée par une classe qui dérive de EJBObject.

Le bean doit implanter l'interface convenant à sa catégorie (SessionBean ou EntityBean) ; ces interfaces définissent des méthodes qui permettent au conteneur de gérer les instances du bean. Par exemple, les méthodes *ejbCreate* ou *ejbRemove* doivent être implantées. [CCC03]
On remarque que les EJB ne sont pas accessible directement par le client, il doit passer par un Proxy.

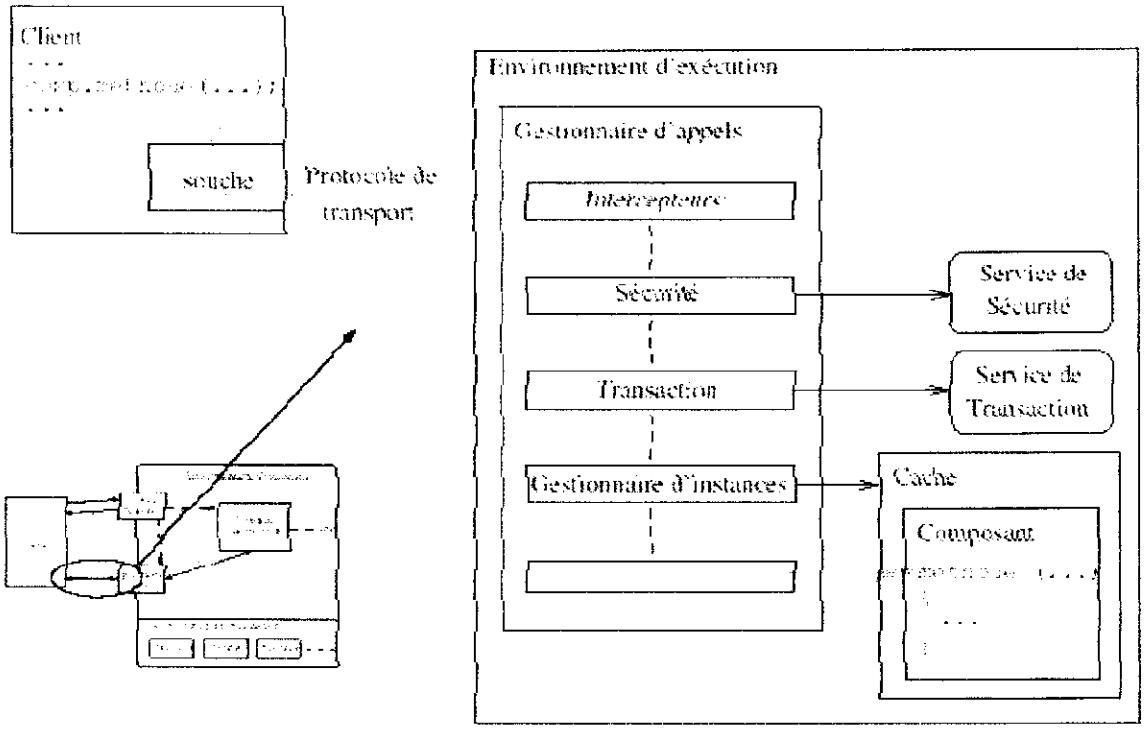


Figure 2.5 : Une requête client[CCC03]

La figure 2.5 : la requête traverse une série d'intercepteurs qui, selon le cas, font appel aux services de l'environnement, par exemple, un service de sécurité bloquera l'appel en début de chaîne. L'appel à la méthode proprement dite du composant se fait en dernier. La réponse suivra le chemin inverse. [CCC03]

V. Empaqueter les EJB :

Les composants EJB sont empaquetés dans un fichier JAR (Java Archive) appelé *ejb-jar*. Ce fichier contient :

II. Les classes Java mettant en œuvre le composant : le bean, la fabrique et le proxy.
III. Le descripteur de déploiement, qui donne les spécifications du composant : catégorie, services du conteneur utilisé, références vers d'autres composants EJB, etc.
IV. Les composants EJB utilisés (par inclusion ou par référence vers un ejb-jar). [CCC03]

VI. Déploiement de composant :

Le composant est défini comme un ensemble de code déployable sur une plateforme d'exécution. L'unité du composant est en général un fichier représentant une archive, contenant principalement le code du composant, et un descripteur de déploiement indiquant comment effectuer ce déploiement. Déployer un composant revient à utiliser le descripteur de déploiement pour :

- connecter le composant avec l'architecture de composant existantant.
- intégrer le code du composant dans l'environnement.
- mettre en place les mécanismes concernant le cycle de vie du composant (selon le type du composant) ;
- mettre en place les services requis par le composant, si ces services sont gérés par l'environnement d'exécution.

Les descripteurs de déploiement sont écrits en XML (eXtended Markup Language). Ils contiennent toutes les caractéristiques nécessaires au bon déploiement du composant (catégorie de composant, interfaces, services requis, . . .). [CCC03]

CHAPITRE II :

Introduction au réseau X25

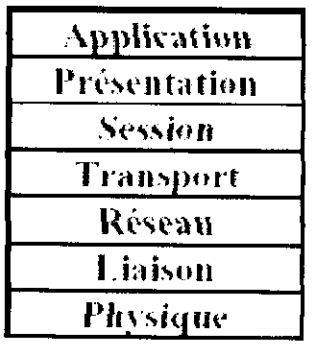
I. Introduction

Le but de ce chapitre est de faire une introduction au X25 et de définir les notions qui seront utilisées dans la suite de cette thèse.
Les points abordés dans ce chapitre sont :

- Rappel sur le model de référence OSI (OPEN SYSTEM INTERCONNECTION) et la couche réseau : le but de cette section est de situer le X25 dans ce model.
- La norme X25 :
- Paquet X25
- Le protocole X25
- Le ticket de taxation

II. Rappel sur le model de référence OSI et la couche réseau

Dans les années 1980, des commissions de normalisation ont défini comment écrire un nouveau réseau propre à interconnecter les machines de différents constructeurs. Mais le réseau mondial OSI n'existe toujours pas. Cependant ce modèle a clarifié les choses en matière de réseau.
Ce modèle a abouti à une représentation en couches qui reste une référence pour tout le monde, même si les réalisations différent quelque peu. [CRM]



Le protocole X25 correspond à la couche réseau du modèle.

La couche réseau assure toutes les fonctionnalités de relaie et d'amélioration de services entre entité de réseau, à savoir : l'adressage, le routage, le contrôle de flux, la détection et correction d'erreurs non réglées par la couche 2.
À ce niveau de l'architecture OSI il s'agit de faire transiter une information complète (un fichier par exemple) d'une machine à une autre à travers un réseau de plusieurs ordinateurs. Il existe deux grandes possibilités pour établir un protocole de niveau réseau : *le mode avec connexion et le mode sans connexion. Le premier cas est celui* adopté dans la norme X25.3 (composante de la norme X25 du CCITT, également norme ISO 8208 et quasi standard international des années 80, utilisé dans le réseau français TRANSPAC), le second est celui du protocole IP du réseau Internet. [CRM]

III. La norme X25 :

- Cette norme a été établie en 1976 par le CCITT pour les réseaux à commutation de paquets sur proposition de 5 pays qui l'utilisent pour leurs réseaux publics de communication : **Transpac** pour la France, **EPSS** pour la Grande-Bretagne, **Datapac** pour le Canada, **Telenet** pour les USA [CRM] et **DZPAC** et **MEGAPAC** pour Algérie l'Algérie.

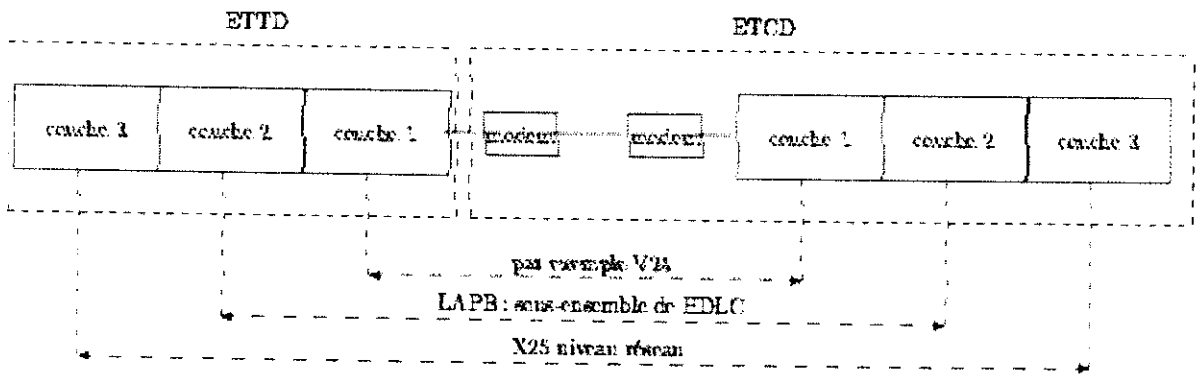


Figure 3.1: Niveaux du protocole X25. [CRM]

Comme illustré dans la figure 1.1 le protocole X25 contient les trois premières couches du modèle OSI et définit l'interface entre un ETTD (Équipement Terminal de Traitement de Données) et un ETCD (Équipement de Terminaison de Circuit de Données) pour la *transmission de paquets. Elle fixe donc les règles de fonctionnement entre* un équipement informatique connecté à un réseau et le réseau lui-même. [CRM].

IV. Paquet X25 :

Le figure représente la format d'un paquet X25 (chaque ligne correspond à un octet)

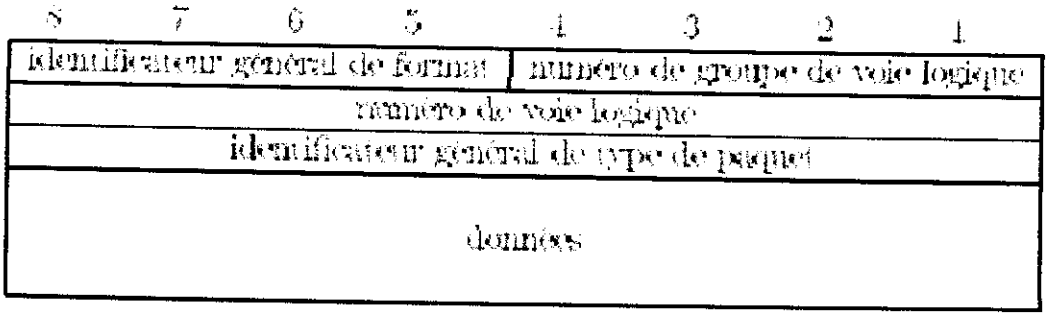


Figure 3.2: Paquet X25. [CRM]

La taille du paquet X25 va évoluer vers 1024 puis 2048 octets

V. Le protocole X25 :

X25 est un protocole de transport d'information complet qui gère le transport de l'information de bout en bout sur de très longues distances avec un plan de numérotation international.

Pendant les années 80, X25 a été beaucoup utilisé, mais sa complexité le rend mal adapté aux hauts débits, aux transports sur fibre optique et souffre de la concurrence de ATM et de Frame Relay.

Des réseaux nationaux utilisent X25 pour le transport des données informatiques. TRANSPAC en France est l'opérateur national. Celui-ci facture le service comme pour le téléphone et on est facturé à la durée de la connexion et aussi aux nombres de paquets X25 transportés (mais pas à la distance sauf international).
X25 définit donc un réseau téléphonique pour ordinateur.
On définit deux types de machines, les DTE et les DCE. Le DTE est un terminal ou un ordinateur, alors que le DCE est un modem, ou un commutateur X25.

- L'ordinateur compose un numéro qui va appeler un autre ordinateur. L'appelé peut refuser la communication, lire des données complémentaires du paquet d'appel de manière classique,
- Le DTE initie un appel via un numéro (175xxxxx pour paris..).
- Le réseau route ce paquet d'appel et crée ce que l'on appelle un Circuit Virtuel.

Ce protocole est orienté connexion, c'est à dire que tous les équipements le long de la ligne vont garder la mémoire de ce chemin et réserver des ressources (mémoire sous forme de buffers et de files d'attente).
Ce système permet une connexion avec un temps de réponse garantie et un contrôle d'erreur au niveau de chaque liaison.
Autre avantage, les paquets ne transmettent pas l'adresse du destinataire une fois le CV effectué. Seuls des numéros de voies logiques sont transmis entre le point d'appel et le premier commutateur.
Par contre dès qu'une ligne a un incident, le CV est coupé, les sessions sont perdues. Il faut se reconnecter.
Chaque liaison entre commutateurs X25 est basée sur les trames de niveau 2 HDLC/LAPB. Chaque commutateur ne peut supporter qu'un nombre restreint de CVs (Circuit Virtuels) et ceci même si les liaisons ne véhiculent pas de données.
Grâce à un plan de numérotage de type téléphonique, les commutateurs n'ont pas besoin de protocoles de routage complexe.
En fait avec quelques centaines d'adresses, le commutateur peut travailler de manière autonome.

X25 est complexe car il mélange à la fois des problèmes de réseau (router l'information) et des problèmes de transport. Par rapport aux couches TCP/IP, on pourrait dire que X25, c'est IP+TCP+OSPF+ICMP Bref c'est du niveau 3++.
Avec ATM ce sera pire, car ATM se préoccupe d'être universel et veut aussi gérer les réseaux locaux, la téléphonie, la vidéo. X25 n'a jamais eu cette prétention pour la bonne raison qu'à cette époque, les réseaux locaux n'existaient pas. X25 a été normalisé entre 1981 et 1984.
Le service LVR sert à interconnecter des réseaux locaux. [LRI]

Sachant que la connexion est réalisée à l'aide d'un circuit virtuel dont le fonctionnement se déroule comme illustré dans la figure 1.3

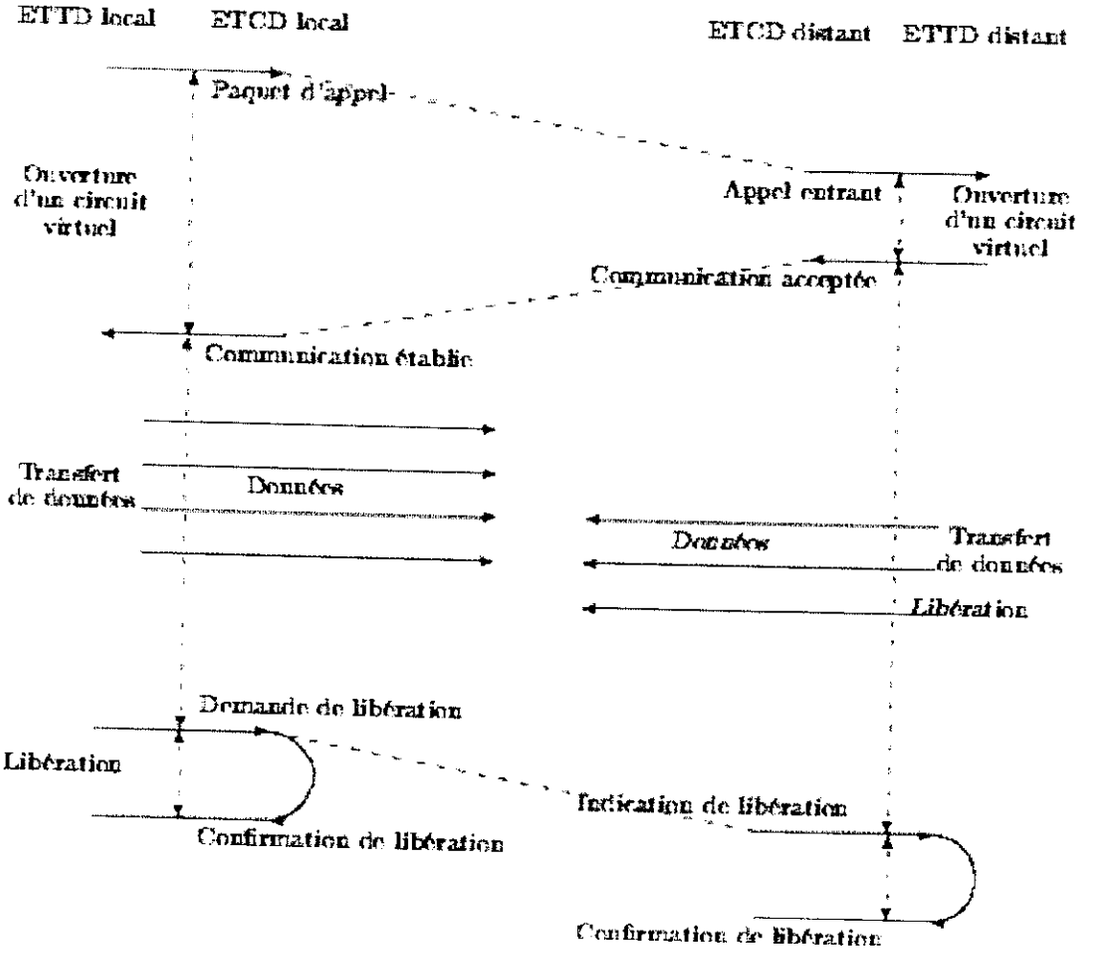


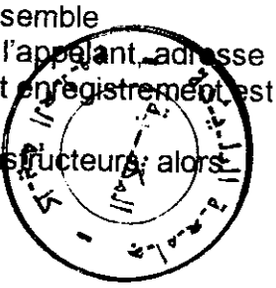
Figure 3.3: Vie d'un circuit virtuel [CRM]

L'établissement du circuit virtuel se fait de bout en bout par l'envoi d'un paquet d'appel (call request). Il existe des circuits virtuels permanents (CVP) ou commutés (CVC). À un instant donné, plusieurs CVC et CVP peuvent coexister. Chaque circuit virtuel utilise une voie logique repérée par un numéro de groupe logique (entre 0 et 16) et un numéro de voie (entre 0 et 255). Ainsi, 4095 (la voie 0 est réservée) voies sont utilisables par une

entrée. Pour un CVC les deux numéros sont attribués pendant la phase d'établissement de la communication, pour un CVP ils le sont lors de l'abonnement. Lorsque le paquet d'appel arrive à l'ETCD destinataire, celui-ci peut refuser la connexion en envoyant une demande de libération (clear request) ou l'accepter en envoyant un paquet de communication acceptée (call accepted). À ce moment-là le circuit virtuel est établi et les données peuvent être échangées, celles-ci emprunteront alors toutes le même chemin marqué à travers le réseau par le paquet d'appel. Pour leur part la demande et la confirmation de libération sont traitées localement et la demande peut être traitée par n'importe lequel des deux équipements. [CRM]

VI. Le ticket de taxation :

Quand deux clients communiquent il y aurait la génération d'un ensemble d'enregistrements qui décrivent cette communication (adresse de l'appelant, adresse de l'appelé, Durée de communication, Volume échangé, ... etc. Cet enregistrement est appelé ticket de taxation (**tt**).
Rappelant que Le réseau X25 peut être construit par plusieurs constructeurs; alors chaque constructeur défini son propre forma de ticket.
L'exemple suivant décrit un **tt** d'un des constructeurs :



- Numéro d'ordre du ticket : (facultatif) (sur 6 digits)
- Marqueur d'origine : (Com. Nationale, Com. internationale « groupe i»)
- Type de CV : « CVC », « CVP », « CVC secours », « CVC passerelle » (« CVC » pour Sagem)
- Nature du ticket :
  - Ticket ouvert de début ou suite de communication.
  - Ticket ouvert de fin de communication (fin de communication ouverte)
  - ticket unique Sagem (communication complète)
- Date de début du ticket : année (4 digits), mois (2 digits) et jour (2 digits)
- Heure de début du ticket : heure (2 digits) et minute (2 digits)
- Durée du ticket en minutes (4 digits) (*arrondie systématiquement à la minute inférieure*)
- Imputation de taxes : « appelant », « appelé », « autre » (uniquement)
- Adresse d'imputation des taxes (numéro de réseau de l'appelant, numéro de réseau de l'appelé (si PCV)) (sur 10 digits)
- Adresse de l'appelant complète (numéro de réseau + s/adresse) (sur 16 digits)
- Adresse de l'appelé complète (numéro de réseau + s/adresse) (sur 16 digits)
- Nombre d'octets (transmis + reçus) pour ce ticket (en koctets) *(arrondi systématiquement au koctet inférieur*) (sur 6 digits)

Pour récupérer les tickets de taxation du réseau, le constructeur définit une station appelée station de supervision **SVR** qui récolte les **tt** et les sauvegardes dans un fichier.

Ce qu'il faut retenir dans ce chapitre :

- X25 définit un réseau téléphonique pour ordinateur.
- Le protocole X25 utilise le mode **connecté**.
- Dans une communication les paquets envoyés et reçus empreintent le même chemin ce qu'on appelle un circuit virtuel **CV**
- Il y a la possibilité d'avoir un chemin permanant entre deux clients, cela est appelé circuit virtuel permanent **CVP**
- Un CVP est affecté au client lors de l'achat d'un abonnement.
- Lors d'une communication, il y aura la génération d'un ensemble d'enregistrement, chaque enregistrement contient des informations sur cette communication. On appelle cet enregistrement un *ticket de taxation*.
- Le réseau X25 peut être construit par plusieurs constructeurs.
- Chaque constructeur a son propre format de ticket de taxation.
- Chaque constructeur a une station ou deux (La deuxième est une station de secours) pour récolter les tickets de taxations et les mettre dans un fichier, on appelle cette station une station de supervision **SVR.**
- Le X25 est implanté sur le réseau **DZPAC** et **MEGAPAC** en Algérie

CHAPITRE III :

Etude du système en coure

I. Introduction :

Avant d'entamer la conception du système de facturation du réseau X25, nous étudierons le système en cours d'utilisation pour avoir une vue détaillée des traitements nécessaires à la facturation et faire sortir les failles du système.

Nous commencerons cette étude par une description globale du système pour avoir une vue sur les traitements effectués par ce système, Il y aura ensuite une description détaillée de chaque sous-système et en fin nous montrerons les faiblesses de ce système.

II. Représentation générale de l'application :

Le système de facturation en cours a deux constructeurs. Nous rappelons que chaque constructeur à son propre forma de ticket.

Les données en entrée du système sont :

- Les tickets constructeur1
- Les tickets constructeur2
- Les données clients saisis par l'interface utilisateur.
- Le plan tarifaire (tarifs, plages horaires....).
-

Les données en sortie du système sont :

- Impressions des factures des clients (Globales ou Détaillées).
- Mise à la disposition de la comptabilité, d'informations sur le montant des factures et les clients.
- Le journal du système et les alarmes enregistrées.

Le centre de facturation lance le processus de facturation tous les jours pour facturer les clients qui spécifie ce jour comme jour de facturation.

Mode de fonctionnement de Système :

a)Sous système d'Acquisition

Le but de ce sous-système est de transférer les fichiers de tickets des stations de supervision vers le centre de facturation.
Le processus d'acquisition est le suivant :

- **Récupération des fichiers de TT** en provenance de la station de supervision du réseau X25 du constructeur1
- **Vérification de l'intégralité du transport.**
- **Effacement des fichiers TT** récupérés sur les deux stations de supervision du constructeur1.
- **Récupération des fichiers de TT** en provenance de la station de supervision du réseau X25 du constructeur2.
- **Vérification de l'intégralité du transport.**
- **Effacement des fichiers TT** récupérés sur les deux stations de supervision du constructeur2.

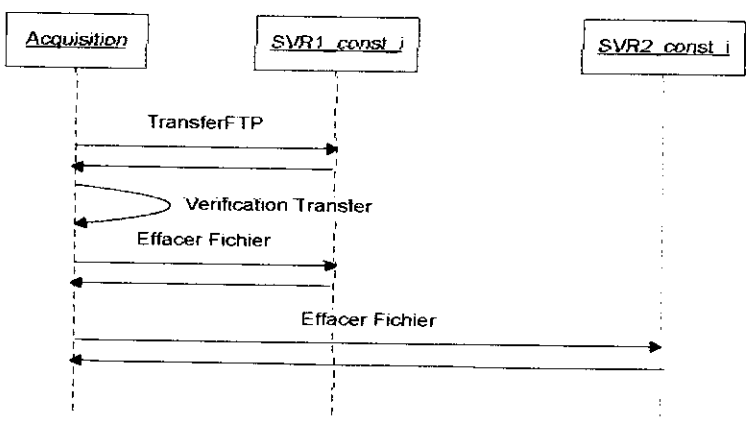


Figure 4.1 : Diagramme de séquence de l'acquisition

b) Sous système de Préparation

Le but de ce sous-système est de générer un fichier de tickets qui ont le même format (Fichier de ticket harmonisé) à partir des fichiers acquis.

La grammaire est un fichier qui indique la position des champs utiles pour la facturation dans un ticket, le caractère de séparation entre les champs et le type de codage (ASCI,...).

Le processus de préparation des tickets du constructeur i (i=1..2) est le suivant :

1. Lire les fichiers des tickets acquis et non préparés.
2. Lire la grammaire du constructeur i.
3. Décodage des tickets : Extraction des champs utiles pour la facturation en utilisant la grammaire.
4. Harmonisation des TT : Création des tickets harmonisés, et les sauvegarder dans un fichier.
5. Dédoublonnage des TT : Il se peut que le fichier des tickets contient des doublants de communication à cause d'une erreur sur les SVR.

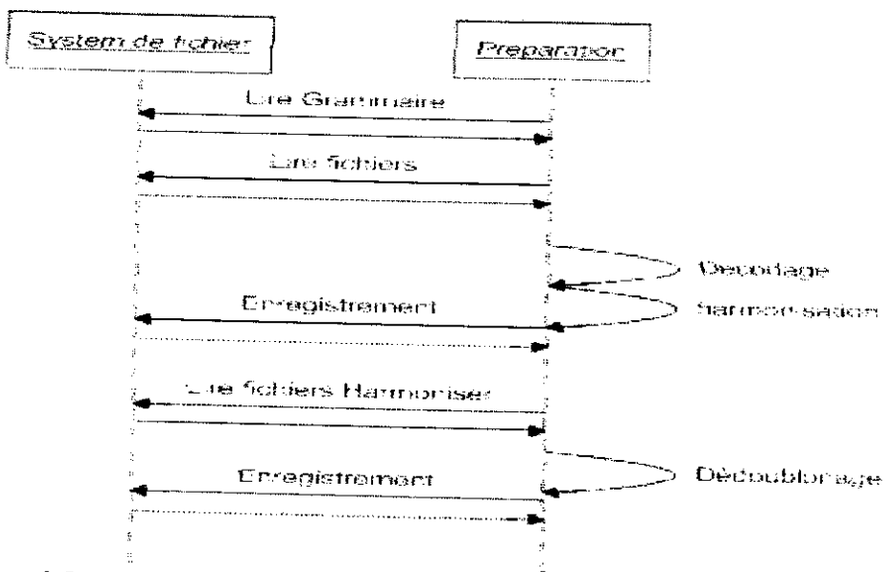


Figure 4.2 : Diagramme de séquence de la préparation

c) Sous-système de Valorisation d'usage :

Le but de la valorisation d'usage est de valoriser l'usage de communication de chaque ticket. Pour calculer ces coûts on a besoin du ticket et des tarifs des communications.
Le processus de valorisation d'usage est le suivant :

1. Enrichissement des TT : Ajouter des nouveaux champs au ticket harmonisé(Code Groupe, Volume sur la tranche 1, Durée sur la tranche 1, ...)
2. Valorisation des TT : Le Valorisation d'usage de chaque ticket.
3. réconciliation et Stockage des données valorisées : Regrouper les tickets partiels en une seule communication et les stocker dans une base de données.

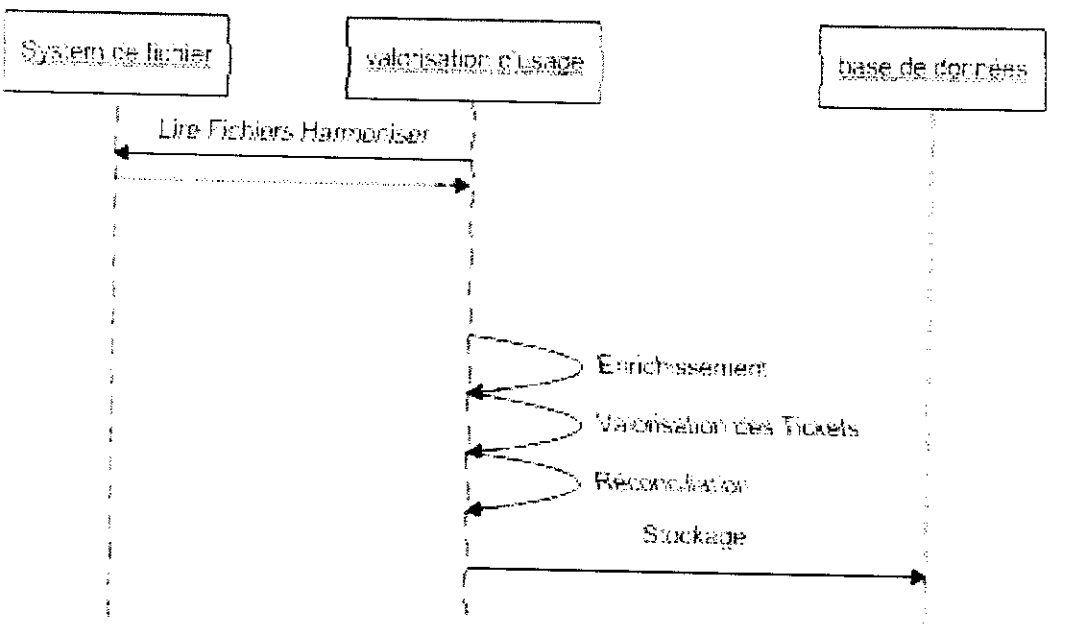


Figure 4.3 : Diagramme de séquence de la valorisation d'usage

d) Valorisation de produit :

- La valorisation de produit est l'étape précédant la facturation. Elle est indépendante de la valorisation d'usage et donc des tickets de taxation. Elle prend en compte les caractéristiques des accès du client. Elle détermine notamment s'il faut éditer une facture pour un client donné et quelle est la période prise en compte pour la facturation. Elle prépare les données pour la facturation des accès des clients à facturer.

e) Calcul de la facturation

Le calcul de la facturation est basé sur :

- le résultat de la valorisation de produit (valorisation des accès),
- le résultat de la valorisation d'usage (valorisation des communications),

Le calcul de la facturation aboutit à :

1. la réalisation d'un flux de données pour la génération d'un fichier pdf correspondant à la facture détaillée dans la langue choisie par le client à facturer. Ce fichier contient les données à faire apparaître dans la facture globale

et la facture détaillée du client (en français ou en Arabe). Il sert de point d'entrée à l'édition de la facture sous **Jet Form**.

2. La réalisation d'un fichier comptable : Ce fichier représente l'ensemble des factures produites par la facturation avec autant de lignes que de clients traités d'une journée de facturation. Ces lignes sont constituées par des informations principales du client y compris le montant de la facture.

III. Les sous-systèmes à étudier :

Nous avons choisi de concentrer notre étude sur le chemin parcouru par les tickets de la station de supervision jusqu'à la base de données, et de revoir l'interface utilisateur qui permet la saisi des différents clients, abonnements, réductions, et tarifs.

IV. Description détaillée du système :

I. Sous-système *acquisition :*

La récupération des tickets est réalisée à la demande du Centre de Facturation. La fréquence de fonctionnement de ce processus est de quelques heures.
On ne peut pas transférer directement le fichier des tickets car il est en cours d'édition.

La fonction SVCTL

Svctl est une fonction fournie par le constructeur qui copie les tickets de taxation du fichier au fil de l'eau sans fermer celui-ci et recopie ces tickets dans le fichier image. Les tickets une fois copiés, sont supprimés du fichier de départ. Si bien qu'au prochain appel de la fonction svctl, on ne peut pas récupérer les mêmes tickets.

En partant de l'hypothèse que le fichier «image» s'appelle «TT.const1», et que le paramétrage du centre de facturation impose la station1 comme station principale et la station2 comme station de secours, la séquence standard des commandes automatiques du centre de facturation vers les SVR est la suivante :

1. Si existence d'un fichier TT.const1 dans l'arborescence $HOME/exploit aller directement au point 5
2. Copie des tickets du fichier de stockage des tickets au fil de l'eau dans le fichier TT. const1 sur la *station1* (en utilisant la fonction « **svctl** » et la commande « **rsh** »).
Remarque : les tickets copiés sont automatiquement effacés du fichier origine par la commande « rsh ».
3. Copie les tickets du fichier de stockage des tickets au fil de l'eau dans le fichier TT.const1 sur la station2 (en utilisant la fonction « **svctl** » et la commande « **rsh** »).
Remarque : les tickets copiés sont automatiquement effacés du fichier origine par la commande « rch ».
4. Contrôler qu'une erreur ne soit pas retournée par l'exécution de ces commandes (voir contrôles et alarmes).
5. Récupérer le nombre d'enregistrement du fichier TT.const1 créé sur la **station1** -> *sum1.*
6. Vérifier le code retour de la commande. (si KO essayer N fois avant de générer l'alarme).
7. *Transfert FTP du fichier TT. const1 contenu dans le répertoire $HOME/exploit* de la **station1**
8. Si le code retour FTP est bon,

Récupérer le nombre d'enregistrement du fichier TT.const1 récupéré -> sum2.
Vérifier que sum1 = sum2.
Sinon
Si le code retour FTP n'est pas bon ou si sum1<>sum2 alors
Tant que nombre d'essais de transfert FTP < T et que le code retour n'est pas bon, relancer le transfert.

Si le transfert n'est toujours pas bon alors créer une alarme mineure et lancer la récupération du fichier créé sur la **station2.**

Fin Si

9. Si un code retour FTP est bon et que sum1=sum2 alors

Lancer la suppression du fichier TT.const1 dans le répertoire $HOME/exploit de la **station2** (commande « rm », voir alarmes).

Lancer la suppression du fichier TT.const1 dans le répertoire $HOME/exploit de la **station1** (commande « rm », voir alarmes).

10. Mise à jour des valeurs de « état de la station » (active, inactive) et « statut de la station » (principale ou secours).

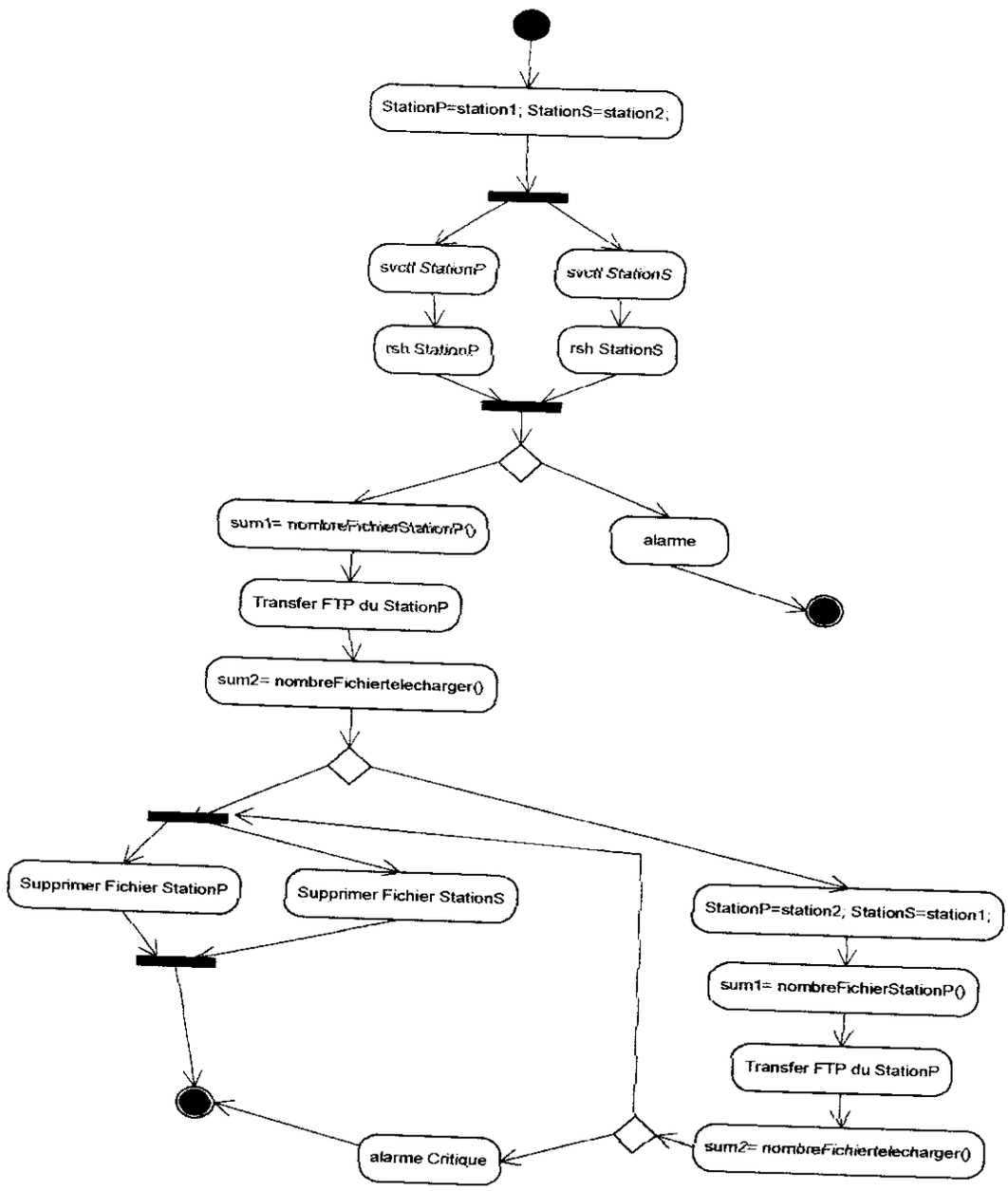


Figure 4.4 : Diagramme d'activité de l'acquisition.

a. Indépendance du sous-système:

Le sous système d'acquisition fonctionne de manière indépendante des autres sous-systèmes.

b. Le Fichier en sortie :

Le fichier récupéré par le sous-système est copié dans un répertoire (paramétré par le responsable système) et renommé par la règle suivante : TT.const1 .AAAAMMJJ.HHMMSS.
Ce répertoire est dédié aux tickets CONST1 (différent du répertoire des tickets Alcatel).
Le sous système suivant (préparation) récupère tous les fichiers de ce répertoire, pour les fusionner et recopie le fichier (unique) dans son répertoire de travail.
Si l'acquisition n'a pas récupéré de fichier, la préparation tournera à vide (ou avec les fichiers éventuellement récupérés par une précédente acquisition sans générer d'alarmes.

II. Sous-système préparation :

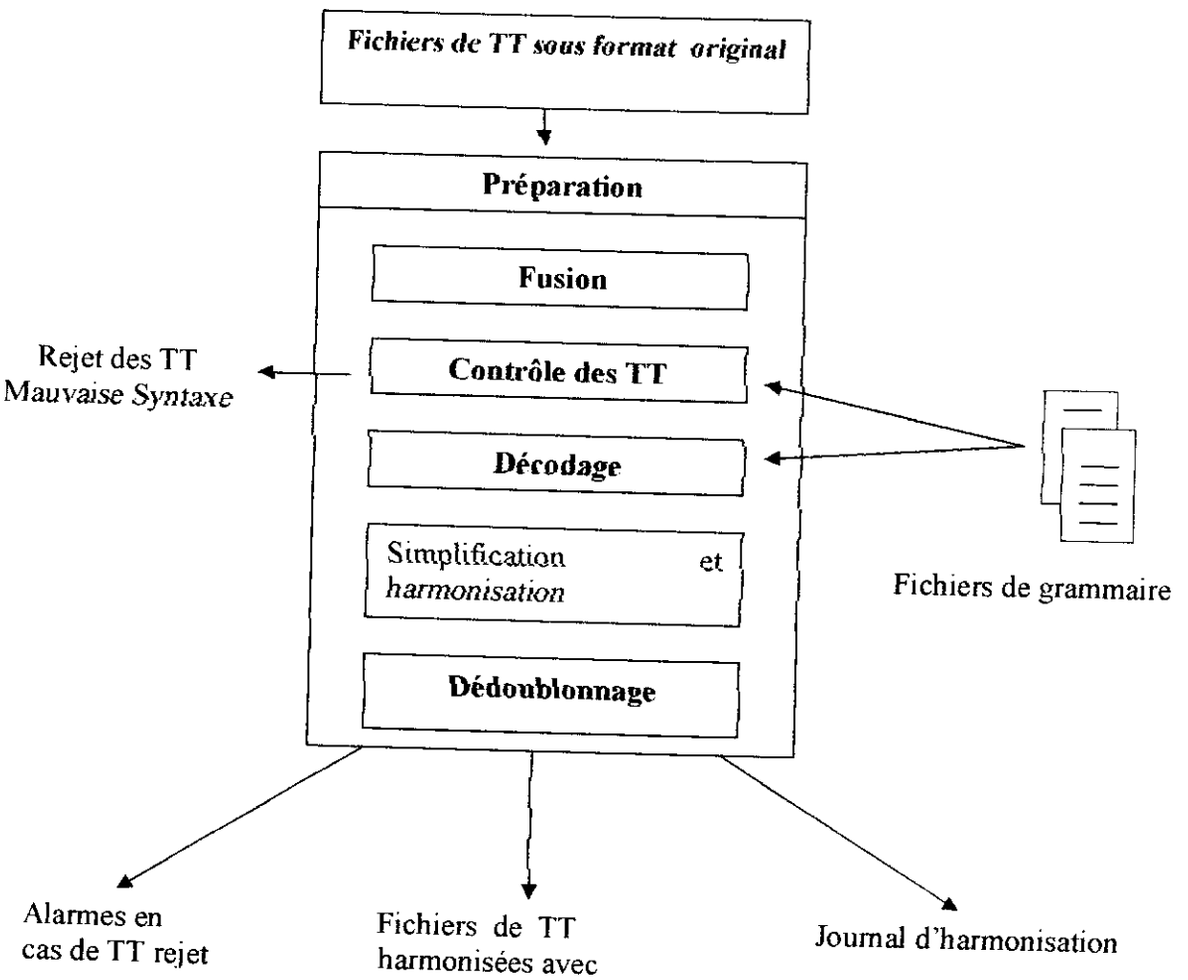


Figure 4.5: Sous-système préparation

Les données en entrée du sous-système de préparation sont les fichiers de TT acquis par la phase d'acquisition. Les fichiers de constructeur 1 sont bien distincts des fichiers de constructeur 2. Les origines de chaque fichier sont connues.

Les fichiers acquis de constructeur N°1 sont fusionnés ensembles par la phase de préparation. Après la fusion les fichiers acquis sont effacés.

Les fichiers acquis de constructeur N°2 sont fusionnés ensembles par la phase de préparation. Après la fusion les fichiers acquis sont effacés.

Les données en sortie sont un fichier unique de TT harmonisés et dédoublonnés avec un indicateur d'origine constructeur N°1 ou constructeur N°2 pour chaque TT.

Chaque TT passe par toutes les phases FUSION, CONTROLE, DECODAGE (SIMPLIFICATION, HARMONISATION et STOCKAGE), DEDOUBLONNAGE. Le sous-système pré - traitement s'appuie sur les grammaires de descriptions de format constructeur N°1 et constructeur N°2.

- La phase de contrôle vérifie la structure syntaxique des TT. Si un TT constructeur N°1 ne possède pas un format reconnaissable par l'application en référence à la grammaire associée, il est rejeté. Il n'y a pas de rejets en ce qui concerne les tickets de constructeur N°2.

- La phase de décodage, simplification, harmonisation transforme les champs codés utiles à l'harmonisation en ASCII, applique les règles d'harmonisation et de simplification et génère un TT de format unique lisible par le sous-système Valorisation d'usage.

- La phase de dé-doublonnage élimine les doublons éventuellement présents dans les fichiers harmonisés.

a. Fusion :

Les fichiers acquis (constructeur N°1 ou constructeur N°2) sont fusionnées ensemble respectivement en un fichier unique relative au premier constructeur et un autre fichier unique relative au deuxième constructeur avant la phase de pré - traitement. Les fichiers acquis sont ainsi effacés.

b. Description des grammaires :

Un champ utile est un champ servant à l'harmonisation directement ou indirectement. Par exemple pour remplir le champ CG_01 du ticket harmonisé à partir d'un ticket on a besoin du champ « Date d'établissement de la connexion du ticket ».
Les champs utiles sont décrits dans les tableaux ci-dessous. Leur position dans les tickets d'origine est définie dans les grammaires.

c. Champs utiles des TT (constructeur N°1) :

Code TT harmonisé	Ticket harmonisé	Code champ utile CONST1	Champs ticket CONST1
CG_01	Date de début du ticket	CUS_03	Date d'établissement de la connexion
CG_02	Heure de début du ticket	CUS_04	Heure d'établissement de la connexion
CG_03	Numéro d'ordre du ticket	-	-
CG_04	Marqueur d'origine	-	-
CG_05	Type de CV	-	-
CG_06	Nature du ticket	CUS_02	Cause – Diagnostic
CG_07	Date de début de communication	CUS_03	Date d'établissement de la connexion
CG_08	Heure de début de	CUS_04	Heure d'établissement de la

	communication		connexion
CG_09	Durée du ticket	CUS_05	Durée de la connexion
CG_10	Imputation de taxes	CUS_02	- Taxation au demandé
CG_11	Adresse d'imputation des taxes	CUS_06 CUS_07	- 9 premiers digits Adresse appelé - 9 premiers digits Adresse appelant
CG_12	Adresse de l'appelant complète	CUS_07	Adresse appelant
CG_13	Adresse de l'appelé complète	CUS_06	Adresse appelé
CG_14	Destination	-	-
CG_15	Nombre d'octets transmis + reçus	CUS_08	Nombre total de paquets transmis et reçus et nombre total d'octets transmis et reçus

CG pour champ générique et CUS pour champ utile

d. Grammaire:

La grammaire est écrite dans un fichier portant un nom spécifique et se trouvant dans un répertoire dédié.
Le séparateur de champ n'est pas forcément unique entre deux champs.
Un champ ne peut être vide.
La ligne de la grammaire est composée : du code défini pour les champs utiles, le nom du champ, le numéro d'ordre du champ et la longueur du champ (vide dans ce cas là car inutile).

```
[CARACTERISTIQUES]
        REGLE_HARMO:↵
        SEPARATEUR:·↵
[CHAMPS]
        CUS_02,Cause - Diagnostic,8,,↵
        CUS_03,Date d'établissement de la connexion,6,,↵
        CUS_04,Heure d'établissement de la connexion,7,,↵
        CUS_05,Durée de la connexion,9,,↵
        CUS_06,Adresse de l'appelé,11,,↵
        CUS_07,Adresse de l'appelant,10,,↵
        CUS_08,Nombre total d'octets transmis,12,,EOF
```

e. Traitement des TT :

Chaque TT subit en premier lieu un contrôle syntaxique global.
Les champs utiles des tickets sont récupérés et décodés.
Et en dernier lieu, les tickets sont harmonisés en un ticket unique à l'aide de règles d'harmonisation et à partir des champs utiles.

f. Contrôle syntaxique global :

Chaque TT subit un contrôle syntaxique global avant d'être décodé et harmonisé. Pour le ticket de constructeur N°1, il s'agit de contrôler que le nombre de champs est égal à 12 et pour le ticket constructeur N°2 il s'agit de contrôler le nombre d'octets du fichier à traiter (longueur utile du ticket est de 256 octets).
En cas de différence, en ce qui concerne le ticket du constructeur N°1, le TT est rejeté tel quel dans un fichier dédié. Une alarme (rajout du code alarme) est générée dans ce cas là. En ce qui concerne le deuxième constructeur, si le fichier d'entrée à une taille incorrecte une alarme est générée, mais le fichier est tout de même traité.

g. Décodage et récupération des champs utiles des TT :

Tous les champs utiles sont en ASCII.

Code du champ	Longueur du champ utile en octet	Nom du champ du ticket CONST1	Valeurs champ utile et commentaire
CUS_02	8 max.	Cause – Diagnostic	1-3/1-3- ou 1-3/1-3R Exemples : 0/0- ou 0/0R ou 255/255-
CUS_03	8 max.	Date d'établissement de la connexion	JJ/MM/A ou JJ/MM/AA ou J/MM/AA ou J/MM/A
CUS_04	5	Heure d'établissement de la connexion	HH:mm
CUS_05	pas de longueur maximum	Durée de la connexion minutes/secondes	1-n/1-n Exemples : 0/0 120/45 5/55
CUS_06	15 max.	Adresse de l'appelé	Exemples : 110100002151515
CUS_07	15 max.	Adresse de l'appelant	Exemples : 110100002151515
CUS_08	pas de longueur maximum	Nombre total de paquets transmis et reçus et nombre total d'octets transmis et reçus	2-n;2-n ;2-n ;2-n

Remarques :

Il faut se référer à la grammaire et aux tickets pour déterminer les champs d'origine et leur emplacement, Aucun contrôle sur les champs n'est effectué à ce niveau là.

h. Harmonisation et règles :

Tous les champs ont des valeurs décimales codées en ASCII

i. Structure du ticket harmonisé :

Champ du ticket	Libellé du champ	Valeurs et Commentaires
CG_01	Date de début du ticket	AAAAMMJJ pour année, mois jour
CG_02	Heure de début du ticket	HHmmss pour heure, minute seconde
CG_03	Numéro d'ordre du champ	000 Champ rempli par la phase de préparation
CG_04	Marqueur d'origine	S pour CONST1 A pour ALCATEL
CG_05	Type de CV	1 pour CVC 2 pour CVP 3 pour CVC secours 4 pour CVC passerelle
CG_06	Nature du ticket	TUA pour ticket unique constructeur N°2 TDA pour ticket début constructeur N°2 TSA pour ticket suite constructeur N°2 TFA pour ticket fin constructeur N°2 TUS pour ticket unique ou fin constructeur N°1 TDS pour ticket suite ou début constructeur N°1 TKR ticket réconcilié (cette valeur est utilisée uniquement au moment de Valorisation d'usage lors de la réconciliation)
CG_07	Date de début de connexion	AAAAMMJJ pour année, mois jour
CG_08	Heure de début de connexion	HHmmss pour heure, minute seconde
CG_09	Durée du ticket	00000 5 digits en minutes (arrondi supérieur)
CG_10	Imputation de taxe	0 pour Appelant pour les appels normaux 1 pour Appelé pour les appels en PCV
CG_11	Adresse de l'imputation de taxe	000000000 9 digits représentant le numéro du réseau
CG_12	Adresse de l'appelant	000000000000000 9 digits représentant le numéro du réseau + 0 à 6 digits maximum représentant l'adresse du sous-réseau.
CG_13	Adresse de l'appelé	000000000000000 15 digit maximum (pouvant contenir l'indicatif international et l'indicatif X121 des pays)
CG_14	Destination	0 pour national 1 pour le groupe 1 2 pour le groupe 2 3 pour le groupe 3 4 pour le groupe 4
CG_15	Volume échangé	00000000 8 digits en octets

g. Harmonisation des champs utiles :

Les champs utiles sont harmonisés en vue d'obtenir les champs génériques du ticket harmonisé.

Les règles sont les suivantes :

Champ du ticket à obtenir	Libellé du champ	Règles d'harmonisation
CG_01	Date de début du ticket	Récupérer la valeur du champ CUS_03 transformé au format AAAAMMJJ
CG_02	Heure de début du ticket	Récupérer la valeur du champ CUS_04 transformé au format HHmmss en considérant ss=0 seconde.
CG_03	Numéro d'ordre du champ	Non rempli
CG_04	Marqueur d'origine	Rempli avec la valeur S
CG_05	Type de CV	1
CG_06	Nature du ticket	TDS si les 7 premiers digits du champ CUS_02 sont égaux à 255/255 TUS si les premier digits du champ CUS_02 différents de 255/255
CG_07	Date de début de connexion	égal au champ CG_01
CG_08	Heure de début de connexion	Egal au champ CG_02.
CG_09	Durée du ticket	Récupérer la valeur du champ CUS_05 au format minutes/secondes et le transformer en minute en arrondissant à la minute supérieure. Si CUS_05 > 99999 minutes alors CG_09 = 99999 minutes.
CG_11	Adresse de l'imputation de taxe	Récupérer les 9 premiers digits de la valeur du champ CUS_06 si le dernier digit du champ CUS_02 est à R Récupérer les 9 premiers digits de la valeur du champ CUS_07 sinon
CG_12	Adresse de l'appelant	Récupérer la valeur du champ CUS_07.
CG_13	Adresse de l'appelé	Récupérer la valeur du champ CUS_06.
CG_10	Imputation de taxe	1 le dernier digit du champ CUS_02 est à R 0 sinon
CG_14	Destination	Non rempli par ce traitement
CG_15	Volume échangé	Somme sur les deux derniers champs de CUS_08 séparés par un point virgule. Champ transformé en Koctet arrondi au Koctet supérieur (1Koctet=1024 octets)

k. Traitement des rejets :

Les TT dont les champs ne sont conformes à la grammaire correspondante sont rejetés dans un fichier dédié stocké dans un répertoire dédié aux TT rejetés.
Les contrôles sont de type vérification de syntaxe, Le tableau suivant donne la synthèse des cas de rejets possibles :

Code rejet	Règle de rejet
PR_S01	Les valeurs JJ MM et A ne sont pas numériques ou le caractère « / » est absent du

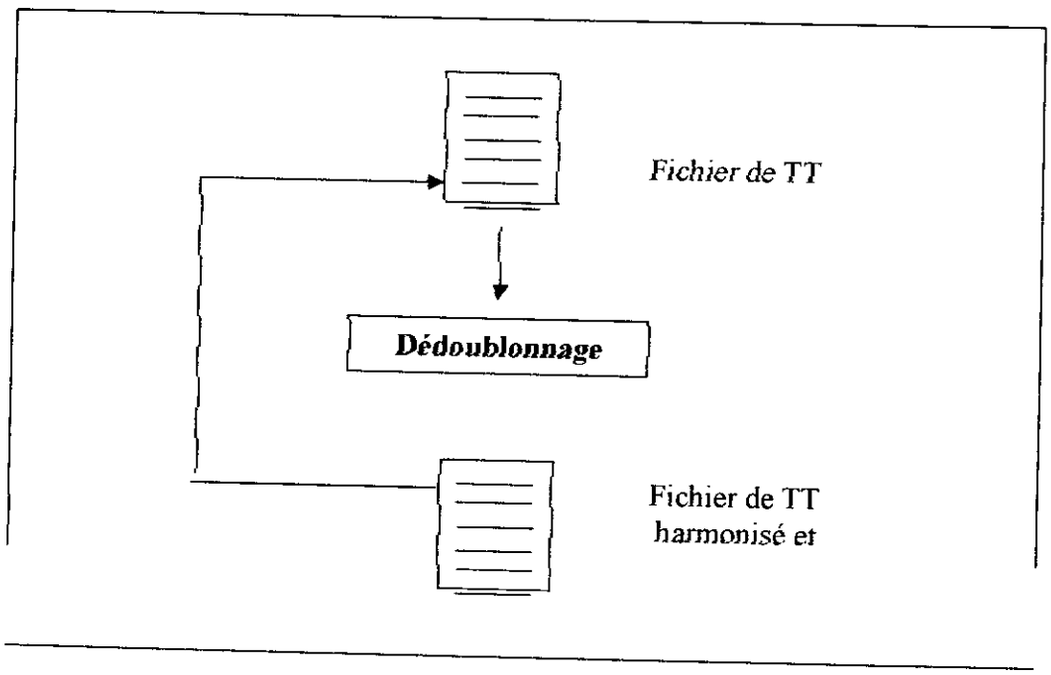
	champ CUS_03.
PR_S02	Les valeurs HH et mm ne sont pas numériques ou le caractère « : » est absent du champ CUS_04.
PR_S03	Le caractère « / » est absent du champ CUS_02.
PR_S04	Les valeurs ne sont pas numériques ou le caractère « / » est absent du champ CUS_05
PR_S05	Au moins une des valeurs à sommer n'est pas numérique ou les 3 caractères « ; » sont absents du champ CUS_08.

I. Stockage des tickets harmonisés :

Les tickets harmonisés sont stockés dans un seul fichier. L'ordre des champs lors du stockage est le suivant et il n'y a pas de séparateur de champ.

Champ du ticket	Libellé du champ	Longueur en octet
CG_01	Date de début du ticket	8
CG_02	Heure de début du ticket	6
CG_03	Numéro d'ordre du champ	3
CG_04	Marqueur d'origine	1
CG_05	Type de CV	1
CG_06	Nature du ticket	3
CG_07	Date de début de connexion	8
CG_08	Heure de début de connexion	6
CG_09	Durée du ticket	5
CG_10	Imputation de taxe	1
CG_11	Adresse de l'imputation de taxe	9
CG_12	Adresse de l'appelant	15
CG_13	Adresse de l'appelé	15
CG_14	Destination	1
CG_15	Volume échangé	8
Retour chariot + Saut de ligne		2
TOTAL		92

m. Dédoublonnage :



Le fichier harmonisé est dé-doublonné avec le fichier harmonisé de la précédente phase de préparation.

III. L'interface utilisateur :

A partir de l'interface utilisateur qui est en cours nous avons extrait les informations suivantes :

- Un client est identifié par un numéro.
- Le centre de facturation applique des réductions sur les communications nationales, des réductions sur les communications internationales et des réductions sur l'accès. Ces réductions sont identifiées chaqu'un par un numéro.
- Chaque client a une réduction sur les communications nationales, une réduction sur les communications internationales et une réduction sur l'accès.
- Chaque client a un ou plusieurs abonnements, chaque abonnement est identifié par un numéro.
- Chaque abonnement a une réduction sur les communications nationales, une réduction sur les communications internationales et une réduction sur l'accès.
- On affecte à chaque abonnement un accès (adresse X25).
- Un accès est identifié par le numéro X25. Un accès X25 est affecté à un seul abonnement mis en service.
- L'abonnement est mis en service avec un débit, un modem et un type de raccordement dans une date de valeur.
- Chaque abonnement peut avoir un certain nombre de services complémentaires dans une date de valeur.
- Le tarif au volume et le tarif à la minute dépendent du lien de l'appelé, Les liens sont divisés en groupe :
  - Groupe national.
  - Groupe international 1 (Europe).
  - Groupe international 2 (Amérique).
  - Groupe international 3 (Asie).
  - Groupe international 4 (Afrique).
- Chaque groupe international a un ou plusieurs pays. Chaque pays est identifié par un numéro.
- Une journée est subdivisée en tranche horaire. Une tranche est identifiée par une date de début et une date de fin.
- On a trois type de jour (Férié, Ouvrable, ou Ouvert).
- Un jour spécial est un jour Férié ou Ouvrable.
- Le centre de facturation applique des réductions sur les groupes.
- Les réductions sur le groupe international dépendent du groupe international et de la tranche horaire.
- Les réductions sur le groupe national dépendent du groupe national (1), la tranche horaire et du type de jour.
- *Un groupe ne peut avoir que trois réductions.*
- Le centre de facturation applique les tarifs suivants :
  - Tarif de redevance qui dépend du débit, modem et le type de raccordement.

- Tarif de mise en service qui dépend du débit et du type de raccordement.
- Tarif à la minute nationale, tarif forfait complet, tarif forfait sur la durée qui dépendent du groupe national, le débit et la date de valeur.
- Tarif à la minute internationale qui dépend du groupe international, le débit et la date de valeur.
- Tarif au volume qui est caractérisé par un code, le tarif d'un KO, le volume minimum facturé et le tarif d'une communication non aboutie.
- Un tarif au volume est affecté pour chaque groupe dans une date de valeur.

- Le centre de facturation applique une dégressivité au volume. Une dégressivité au volume est une réduction sur le volume échangé.
- Un accès X25 appartient à un et un seul constructeur. Chaque constructeur est identifié par son nom.
- Un constructeur a un ou plusieurs stations de supervision. Chaque station est identifiée par son Host.
- Une communication est identifiée par un numéro.

Le diagramme de classe est le suivant :

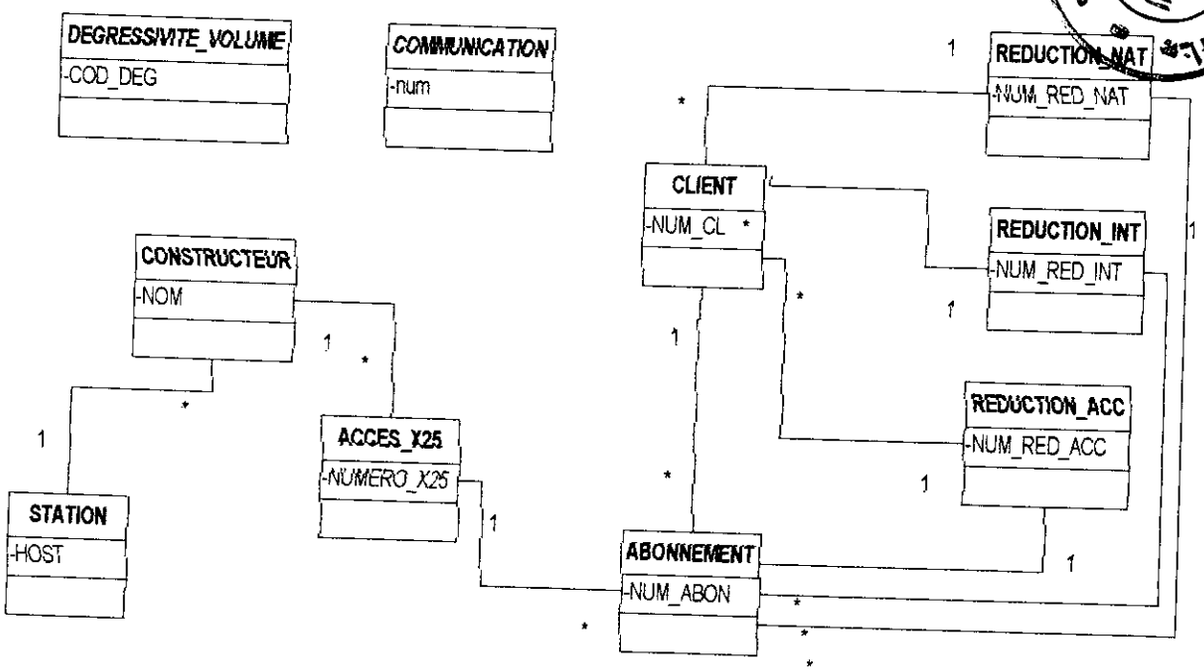


Figure 4.6: Partie1 de diagramme de classe

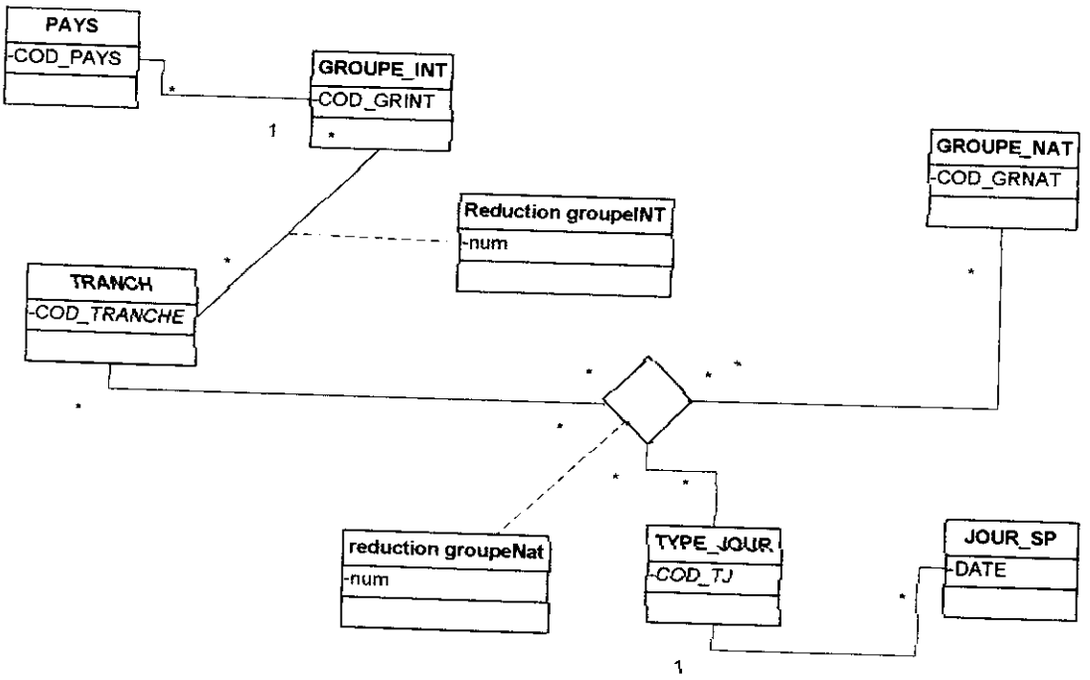


Figure 4.7: Partie2 du diagramme de classe

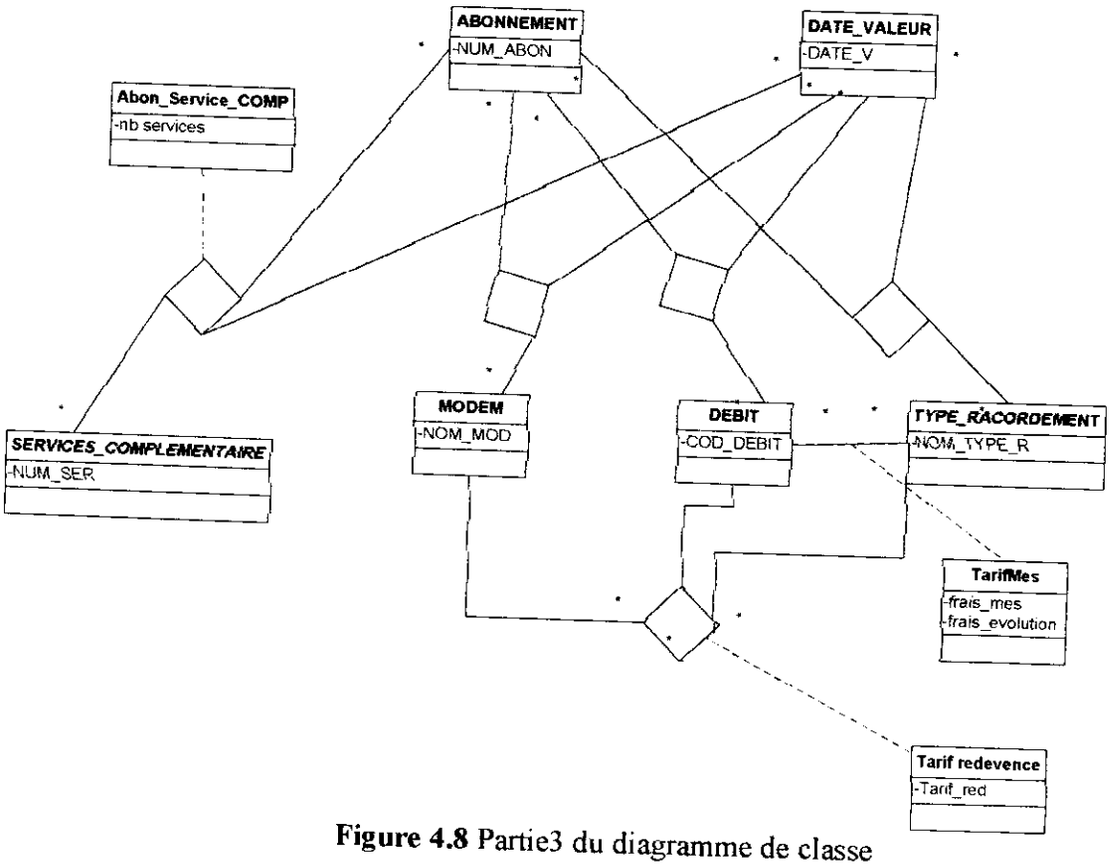


Figure 4.8 Partie3 du diagramme de classe

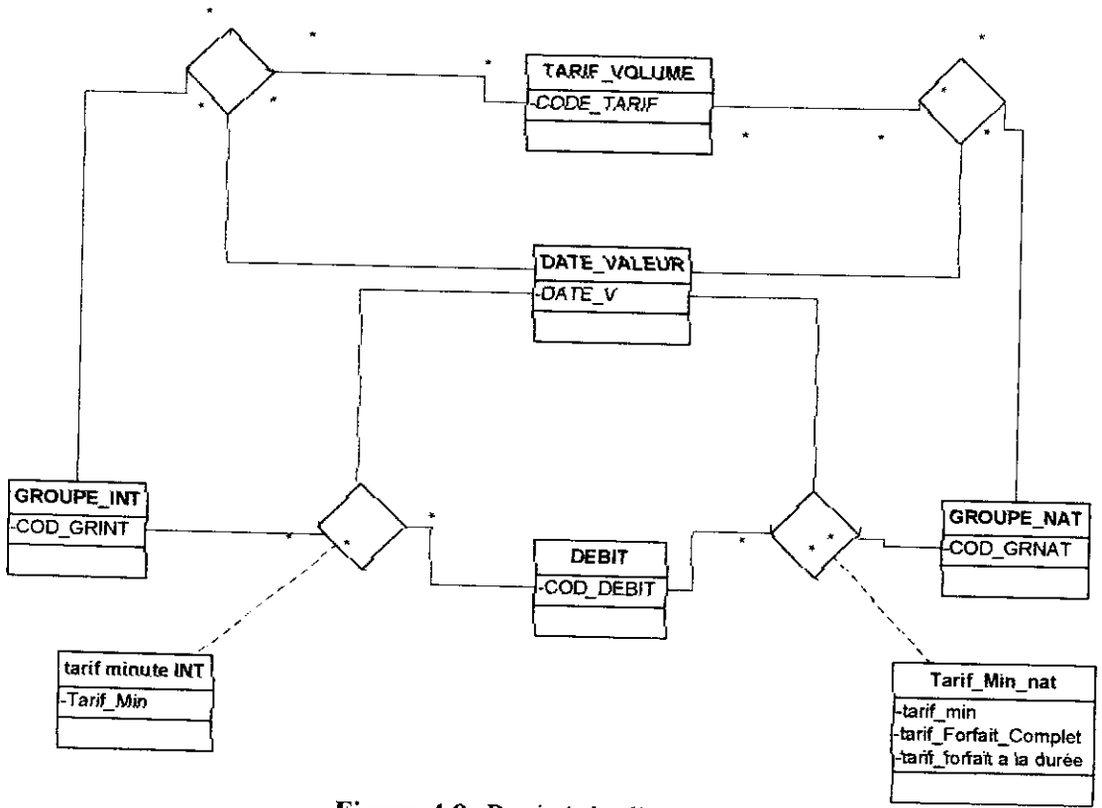


Figure 4.9: Partie4 du diagramme de classe

V. Critique du système en cours :

- Le travail est en mode batch donc on ne peut pas facturer un seul client.
- Le processus d'acquisition utilise les deux stations pour effectuer le transfert, ce qui contredit la notion de station de secours
- Si le processus d'acquisition passe à la station de secours après une panne sur la station principale il y a un risque d'avoir des doubles des tickets de taxation.
- A cause de ses deux dernières faiblesses les responsables ont pris la décision de ne pas utiliser la station de secours. Ce qui implique une isolation du centre de facturation en cas de panne de la station et un très grand fichier des tickets sur SVR car il ne peut être renommé.
- La grammaire utilisée dans la préparation n'est pas suffisante pour extraire les champs à facturer de n'importe qu'elle constructeur, donc le processus de préparation ne marche que pour ces deux constructeurs.
- La suppression des doubles a une complexité de **(n*(n-1))/2** tel que **n** est le nombre de tickets qui est très souvent supérieur à 10 000.
- La valorisation d'usage valorise chaque ticket ce qui nécessite plusieurs accès à la base de données. Pourquoi ne pas rassembler tous les tickets qui décrivent la même communication dans une seule communication et faire un seul calcul ?

CHAPITRE IV :

Proposition de solution

I. Introduction :

Maintenant que les failles du système en cours sont levées nous étudierons notre proposition de solution informatique et nous décrivons l'architecture globale de notre système.

II. Proposition de solution :

Il est clair que les failles de l'ancien système se trouvent dans presque tous ces sous système, alors nous opterons à une refonte de tout le système.

Nous voulons créer une application qui peut être accessible par les clients, alors nous proposons une application basée sur le web.
Donc notre application est une application distribuée accessible par des pages Web

III. L'architecture logicielle :

Nous avons opté à une architecture 3 tiers. Le premier tiers est dédié à la base de données, le second tiers est dédié à la logique métier, et le dernier tiers sera pour le module web.

IV. L'architecture matérielle :

Notre architecture est basée sur un serveur qui va recueillir notre application et des postes clients reliés dans un réseau LAN.

V. L'architecture globale de notre application :

La figure 5.1 montre l'architecture de notre application :

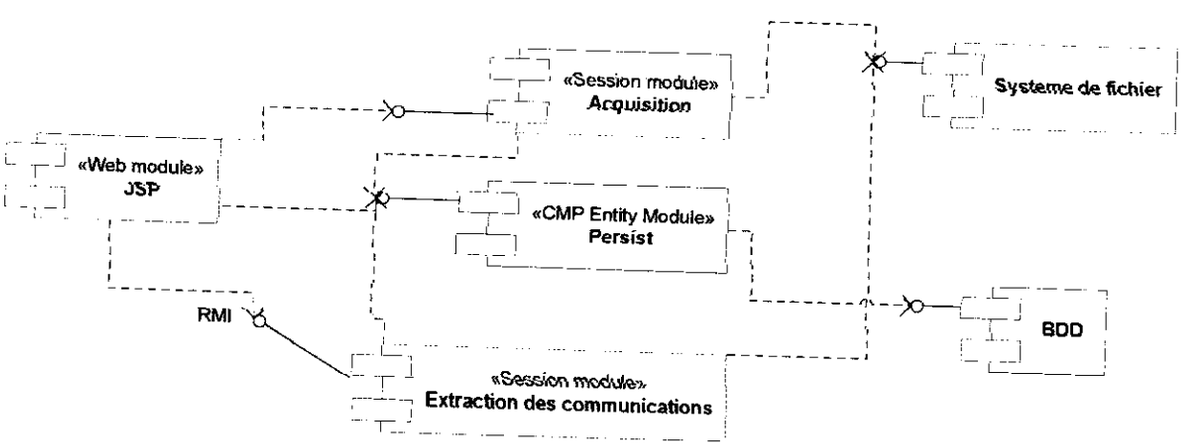


Figure 5.1: l'architecture de l'application

Vérification de notre architecture :

La vérification sera faite au niveau des EJB car ils ont un certain nombre de restrictions. Une des restrictions des EJB est l'interdiction de l'utilisation du système de fichier. Alors que nos deux EJB 'Acquisition' et 'extraction des communications' utilisent le système de fichier.

Pour résoudre ce problème, l'acquisition et l'extraction de communication seront des applications mono poste qui s'exécute dans le serveur. La figure 3.4montre la nouvelle architecture.

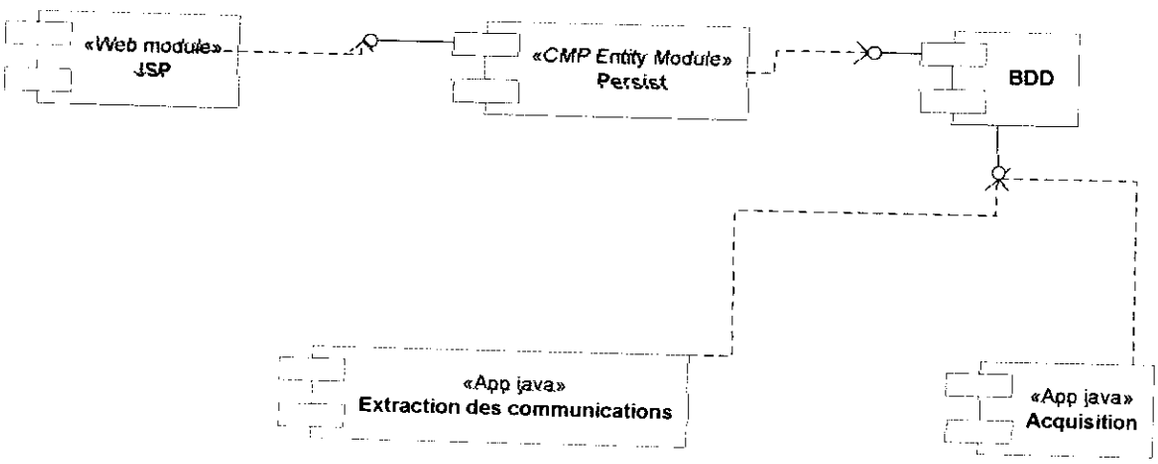


Figure 5.2: l'architecture de l'application version 2

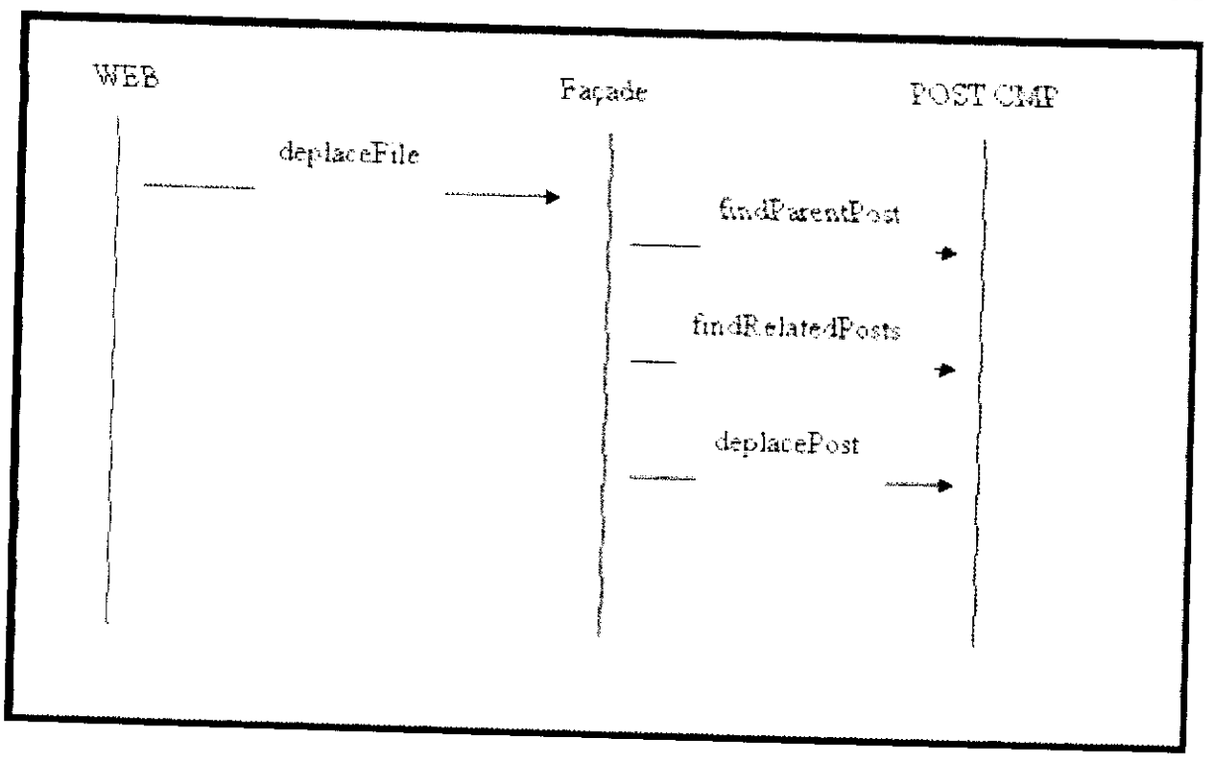
VI. Pattern Session Façade

Chaque appel distant est coûteux, il faut trouver un moyen de les limiter. Une Session Façade est un session bean qui a accès aux interfaces locales d'autres beans (parce qu'il vit dans la même JVM). Un appel à une méthode d'un Session Façade entraîne généralement plusieurs appels vers un ou plusieurs autres beans. Supposons par exemple le cas suivant :

Un modérateur veut déplacer un fil (série de POSTs) d'un FORUM vers un autre. Nous avons ces étapes :

1. Trouver le premier POST du fil à déplacer
2. Trouver tous les POST qui suivent le fil
3. Déplacer chacun des POSTs

Le modérateur ne veut se soucier que du fait qu'il déplace un fil. Notre Session Façade va posséder la méthode deplaceFile qui prend comme argument l'identifiant d'un POST.



Le réseau se trouve entre le serveur WEB (Servlet Container, ou SC) et la Façade. Les appels entre Façade et POST CMP sont locaux. Notre opération ne requiert qu'un seul appel distant, quel que soit le nombre de POSTs à déplacer.

De façon générale, c'est un bon exercice de ne pas générer d'interfaces distantes pour ses EJB persistants. En d'autres termes, une façade porte bien son nom, un mur entre application et persistance.

Notre façade pourrait faire tout un tas d'autres choses, comme par exemple vérifier l'autorisation de l'utilisateur qui a instancier la requête. Une façade peut aussi appeler d'autres Session Beans ou Message Driven Beans. La leçon à retenir étant qu'il faut limiter le nombre d'appels distants. **[01]**

VII. Application du pattern Session Façade sur notre application :

Apres avoir vu les avantages du pattern de façade L'architecture de notre application sera la *suivante* :

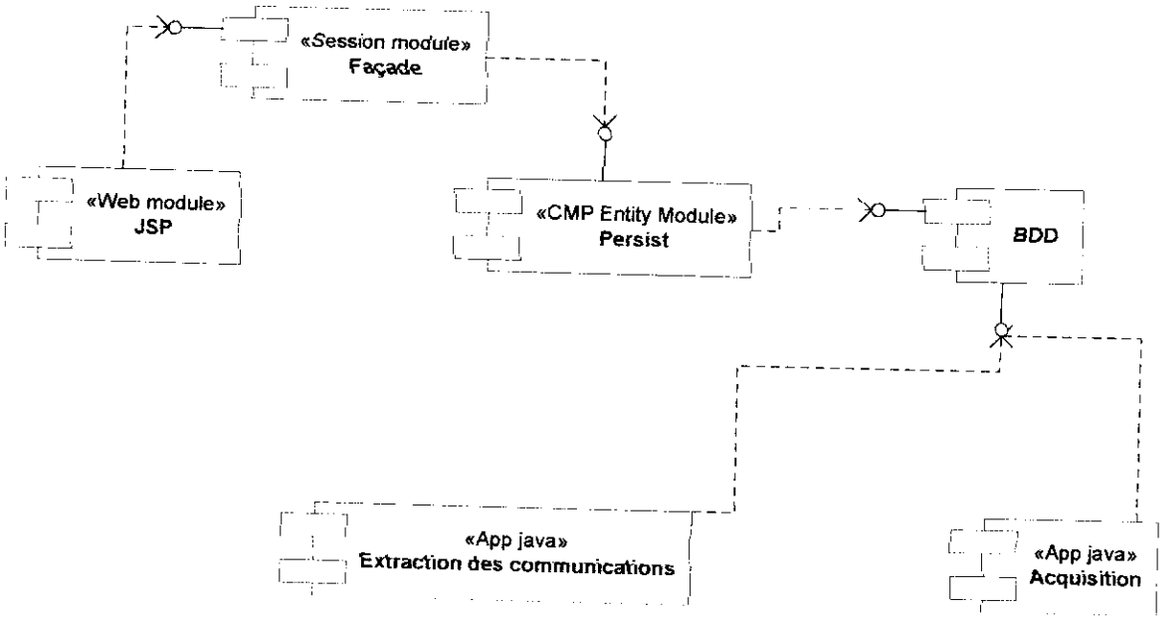


Figure 5.3: l'architecture de l'application version final

CHAPITRE V :

Conception détailler

I. Introduction :

Maintenant que nous avons décrit l'architecture globale de notre système nous entamons dans ce chapitre la conception de ce dernier.
Notre application est divisée en deux parties :

- Une partie monoposte : l'acquisition et le l'extraction des communications.
- Une deuxième partie est distribuée.

II. Conception de l'application acquisition :

Le but de cette application est le téléchargement des fichiers de tickets de la station de supervision vers le centre de facturation.
Pour résoudre les problèmes rencontrés par l'ancien système nous adoptons la solution suivante:

C'est à la station de supervision SVR qui doit renommer le fichier des tickets en utilisant une horloge.

Le Composant acquisition télécharge les fichiers et sauvegarde le numéro du fichier le plus récent pour ne pas télécharger les mêmes fichiers une deuxième fois.
Cette solution nous permet d'avoir la possibilité d'utiliser des stations de secours.

On peut encore optimiser notre solution en ajoutant la règle suivante :
La station de supervision ne renomme pas le fichier des tickets à chaque passage d'un top d'horloge mais chaque fois que le fichier de ticket a un certain nombre de tickets, par exemple 10 000 tickets.
Cette solution garantit que notre système recevra des fichiers de tickets de taille fixe ce qui est un facteur très important dans la suite de notre travail.

Le diagramme de classe de l'acquisition est représenté par la figure.

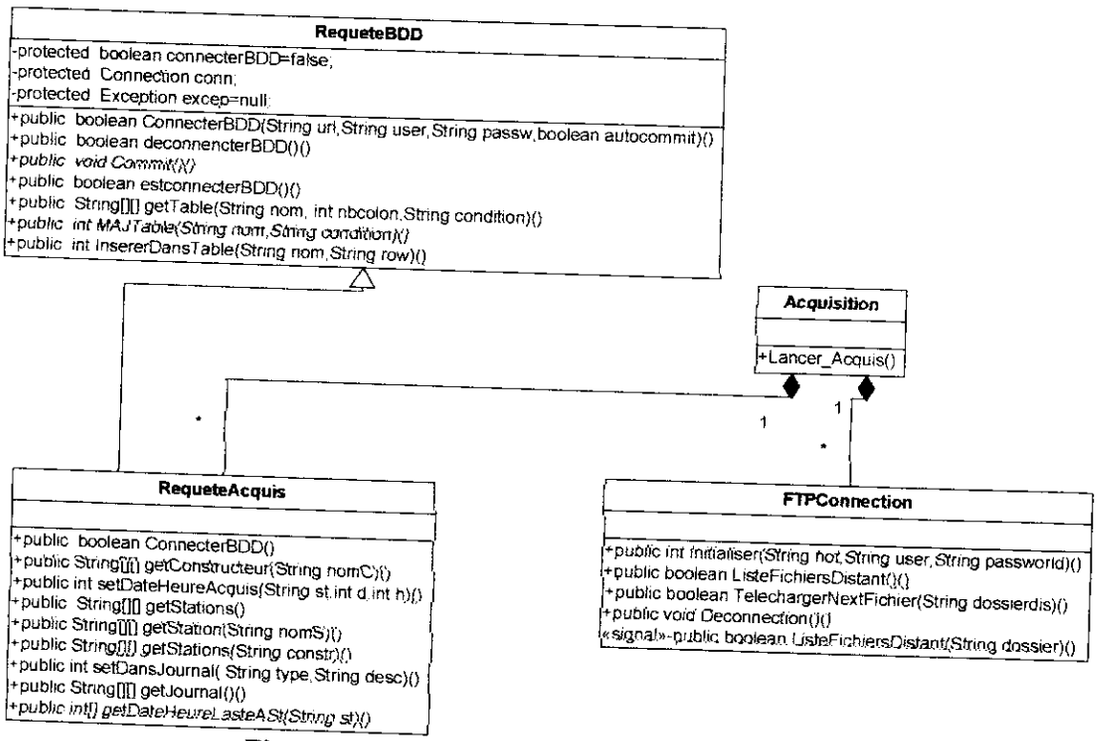


Figure 6.1: Le diagramme de classe de l'acquisition

III. Conception de l'application Extraction des communications

Le but de cette application est l'extraction des communications à partir des tickets de taxation et les mettre dans la Table COMMUNICATIONS.

Pour avoir une application qui extrait les communications de n'importe qu'elle constructeur nous utiliserons la réflexion en java.
Le diagramme de classe de l'application Extraction des communications est représenté dan la figure :

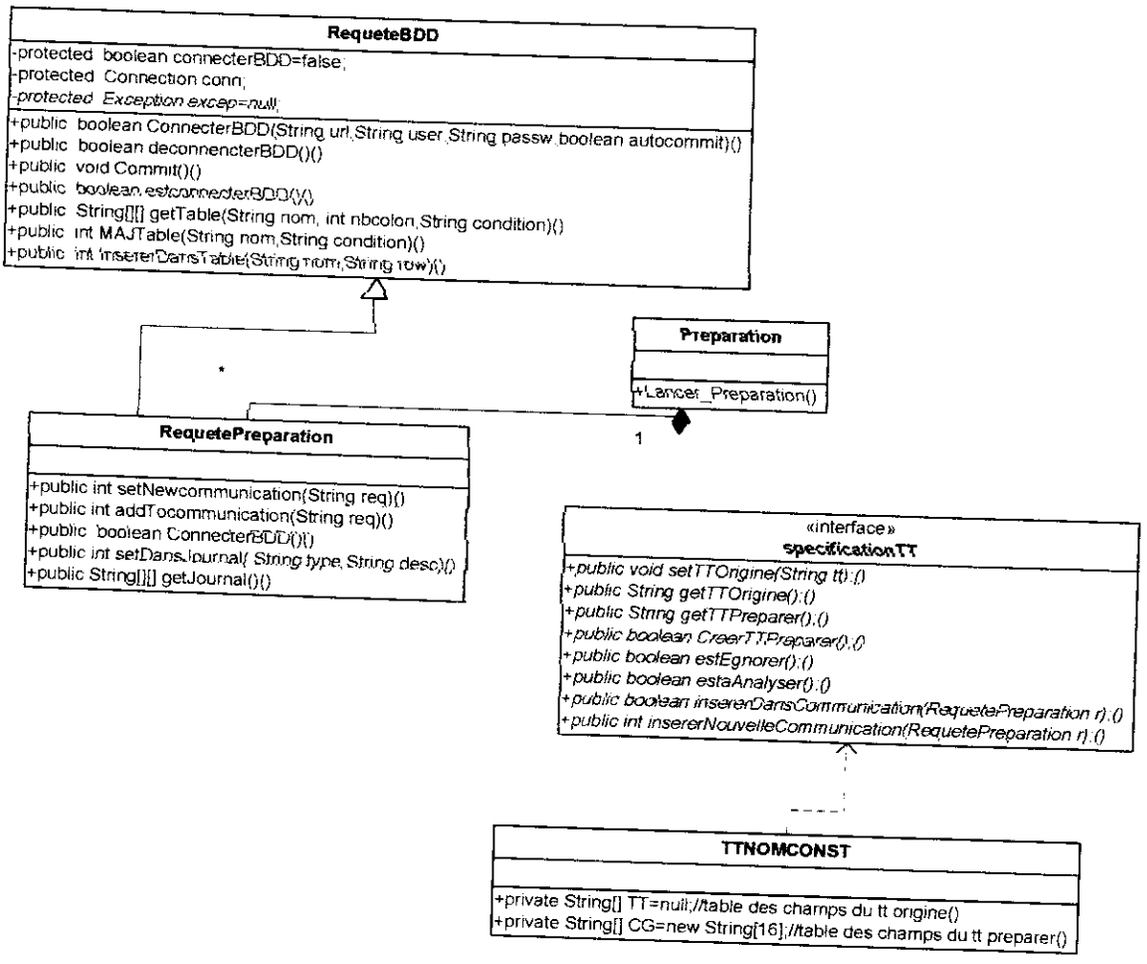


Figure 6.2: Le diagramme de classe de l'extraction des communications

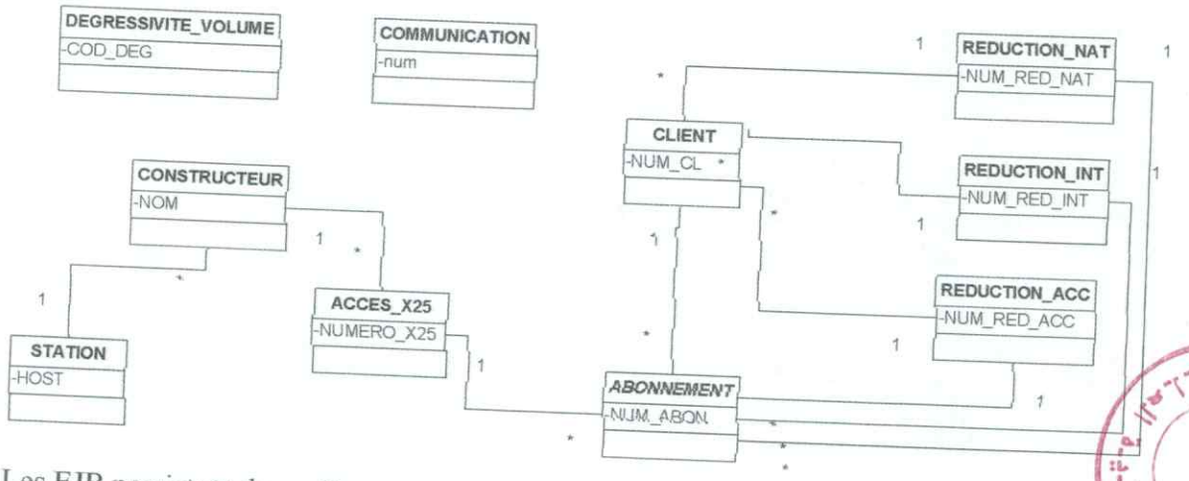
IV. Conception des EJB :

1) Le module persist

i. Extraction des *EJB* persistants :

A partir du diagramme de classe établi dans l'étude du système. Nous extractons les EJB persistants.

Partie : clients, abonnement, Accès



Les EJB persistant de ce diagramme seront :

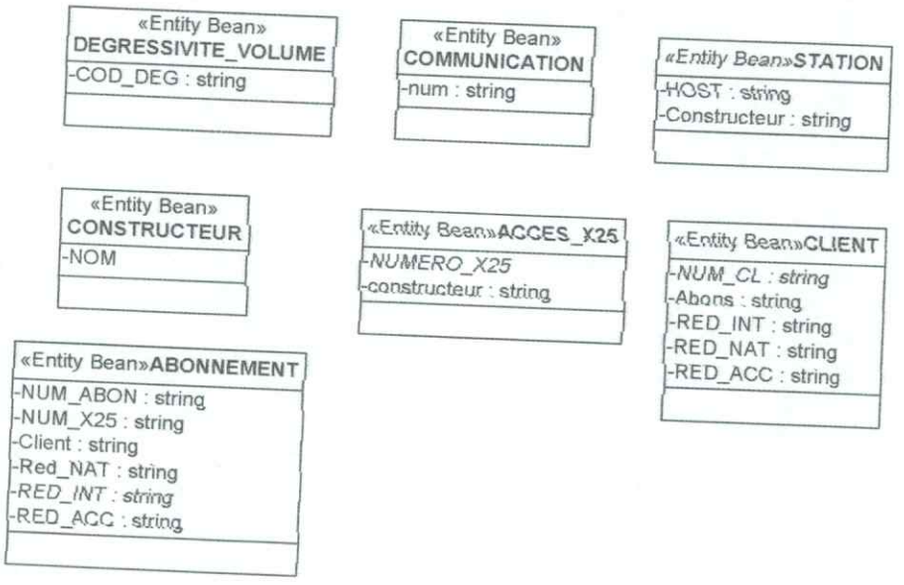
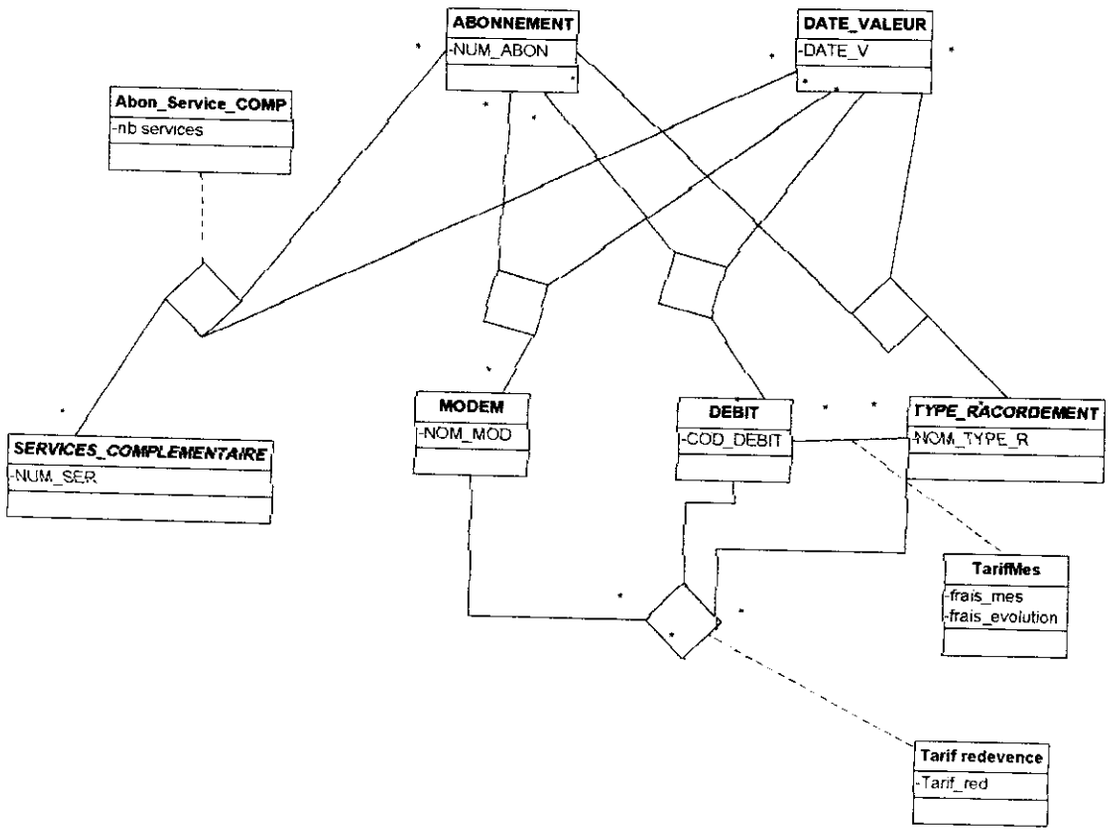


Figure 6.3: Les EJB 1

Partie : clients, abonnement, Acces :



Les EJB persistant de ce diagramme seront :

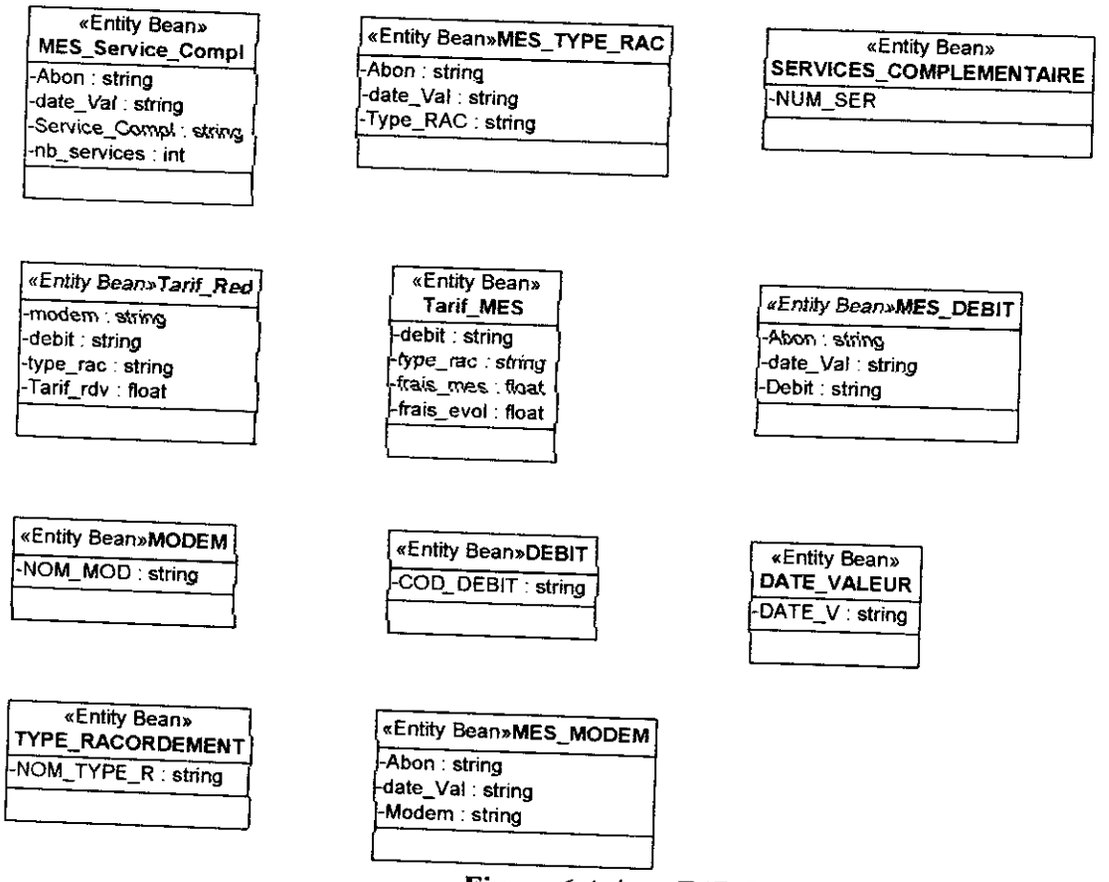
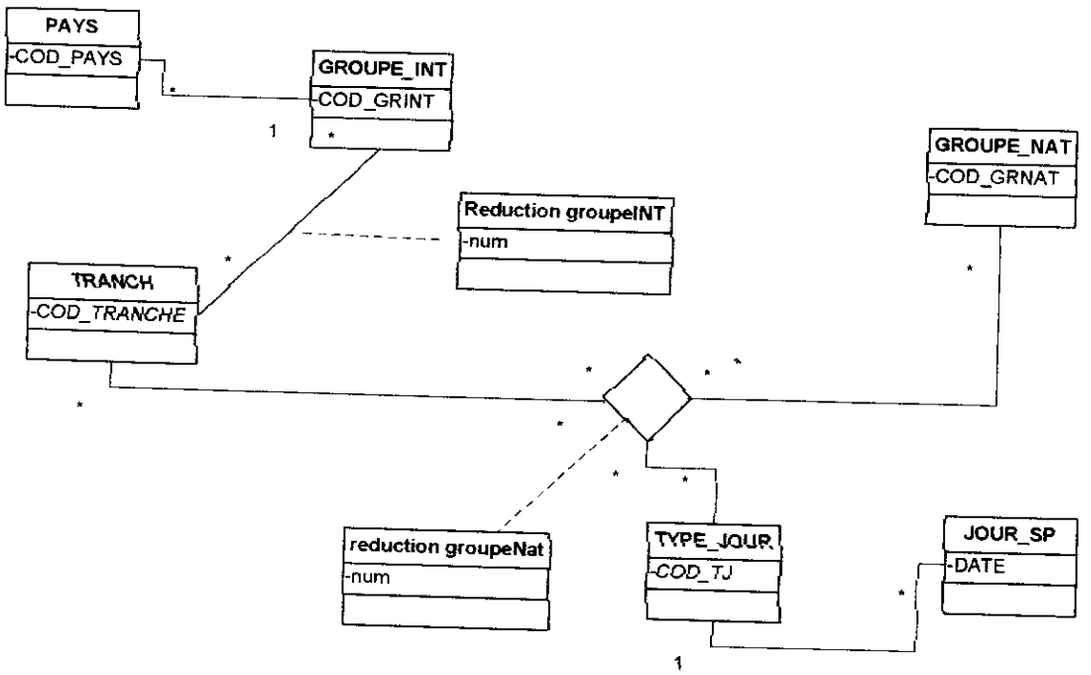


Figure 6.4: Les EJB 2

Partie : Groupes



Les EJB persistant de ce diagramme seront :

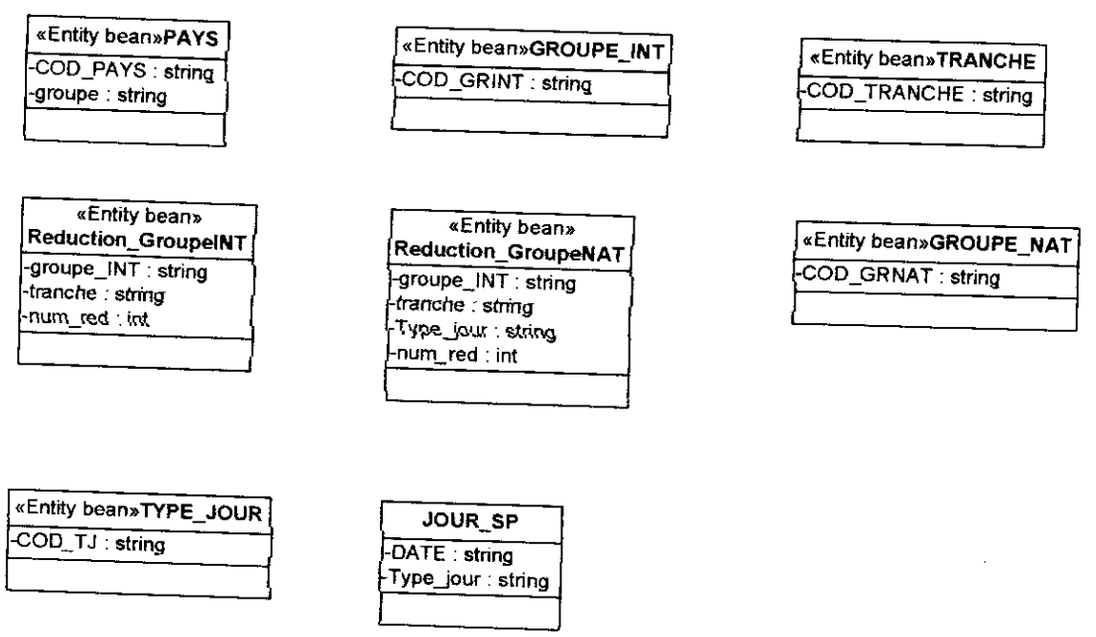
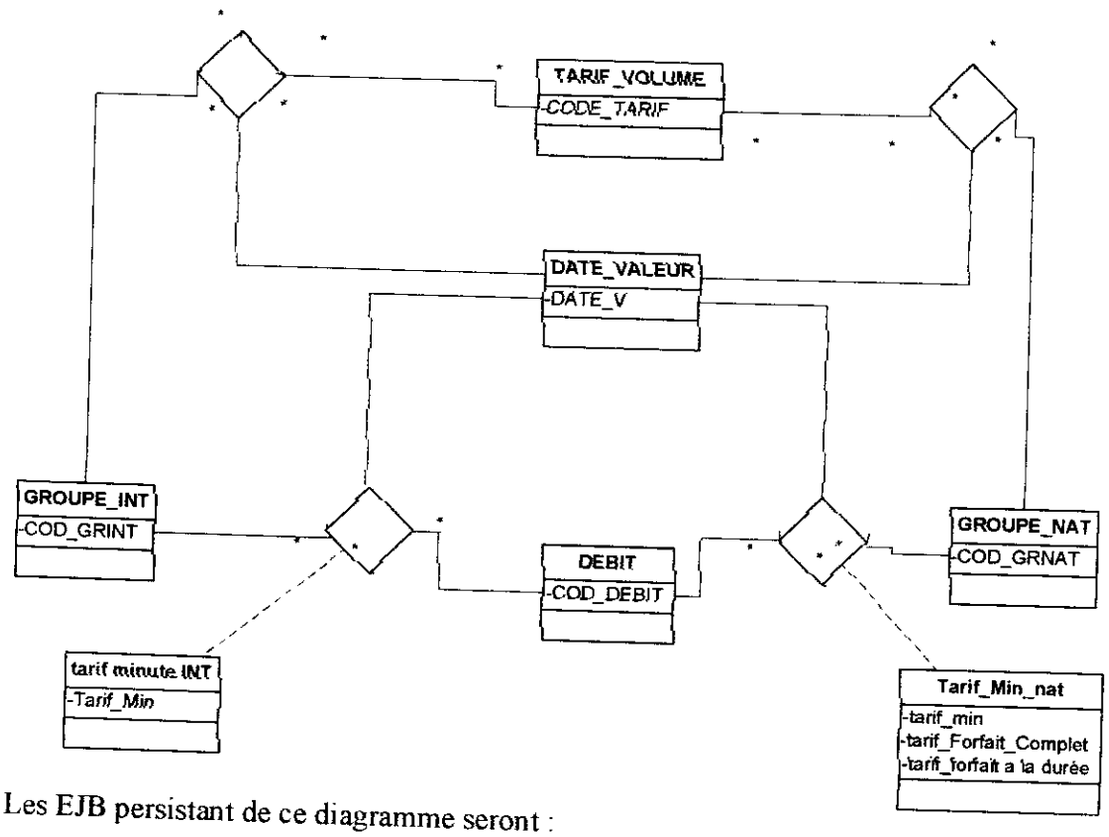


Figure 6.5 : Les EJB 3



Les EJB persistant de ce diagramme seront :

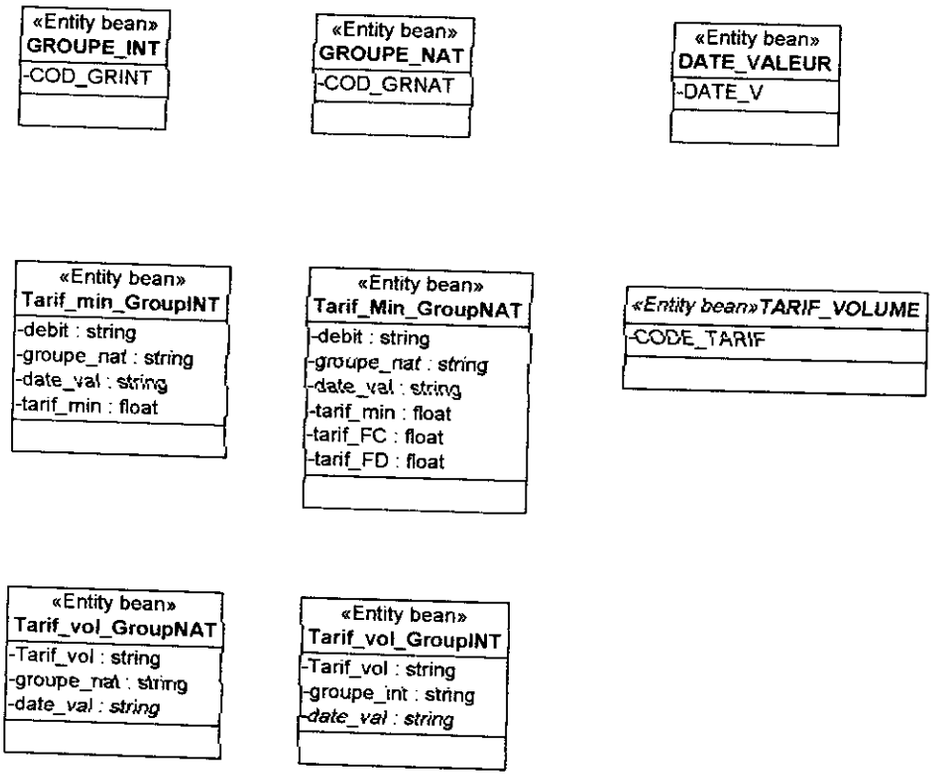


Figure 6.7: Les EJB 4

ii. clés primaires :

La seconde étape consiste à définir la clé primaire de chaque Entity bean :

Entity bean	Clé primaire
TYPE_RACORDEMENT	RAC_COD
MODEME	MOD_COD
DEBIT	DEB_COD
ACCES_X25	ACC_COD
TARIF_REDEVENCE	DEBIT, TYPE_RACORDEMENT, MODEME
TARIF_MES	DEBIT, TYPE_RACORDEMENT,
SERVICE_COMP	SVC_COD
REDUCTION_ACC	REDACC_COD

TARIF_VOLUME	TARIFVOL_COD
DEGRESSIVITE_VOL	DEG_COD
DATE_VALEUR	DATE_V
REDUCTION_INT	REDINT_COD
REDUCTION_NAT	REDNAT_COD
GROUPE_INT	GRPINT_COD
GROUPE_NAT	GRPNAT_COD
PAYS	PAYS_COD
TYPE_JOUR	TYPEJ
JOUR_SP	DATE
TRANCHE	TR_COD
REDUCTION_GROUPENAT	TYPE_JOUR, TRANCHE, GROUPE_NAT
REDUCTION_GROUPEINT	TRANCHE, GROUPE_INT
AFFECTER_TARIF_VOLINT	GROUPE_INT, DATE_VALEUR, TARIF_VOLUME
AFFECTER_TARIF_VOLNAT	GROUPE_NAT, DATE_VALEUR, TARIF_VOLUME
AFFECTER_TARIF_MININT	GROUPE_INT, DATE_VALEUR, DEBIT
AFFECTER_TARIF_MINNAT	GROUPE_NAT, DATE_VALEUR, DEBIT
CLIENT	NUM
ABONNEMENT	ABON_COD
ABON_SERVICE_COMPL	ABONNEMENT, SERVICE_COMP, DATE_VALEUR
ABON_MODEM	ABONNEMENT, MODEM, DATE_VALEUR
ABON_DEBIT	ABONNEMENT, DEBIT, DATE_VALEUR
ABON_TYPE_RACOR	ABONNEMENT, TYPE_RACORDEMENT, DATE_VALEUR
STATION	HOST
CONSTRUCTEUR	NOM_CONS
COMMUNICATION	num

iii. Conception interne :

Chaque EJB Entity doit définir :

- La classe : **[NOMEJB]Bean** qui implémente l'interface *EntityBean*.
- L'interface **Local[NOMEJB]Home** qui hérite *de EJBLocalHome*.
- L'interface **Local[NOMEJB]** qui hérite de *EJBLocalObject*

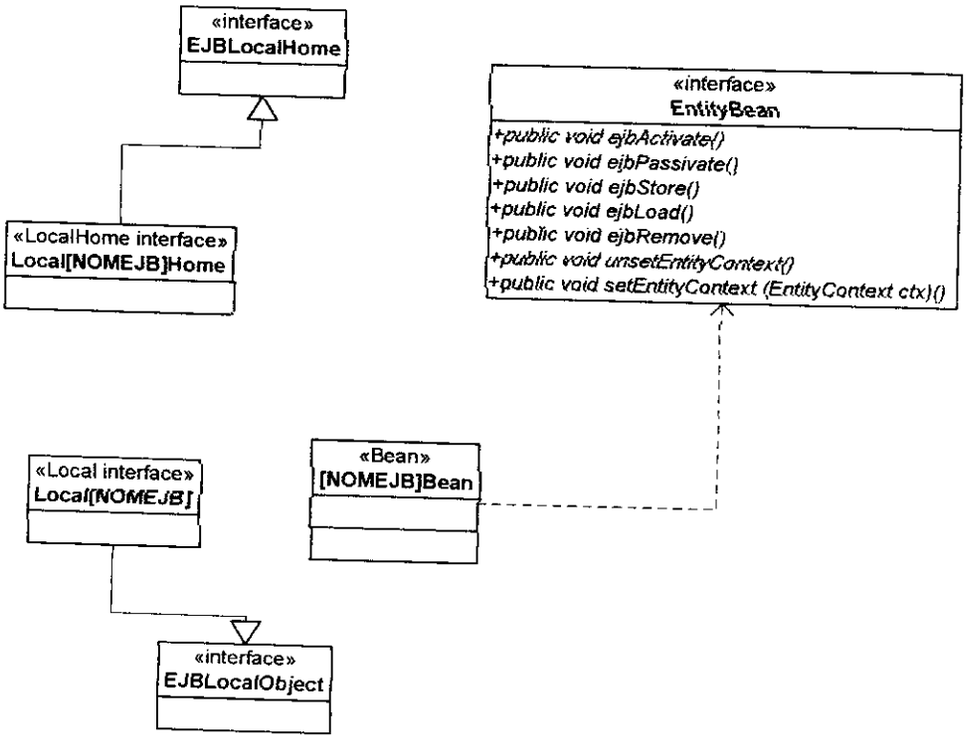


Figure 6.8: La conception interne du EJB Entity

➢ La clé est représentée par un seul attribut :
Nous supposons que le EJB est appelé **NOMEJB** et que la clé est appelée **CLE** de type ***TYPE***.
La conception interne de ce type d'EJB est représentée dans la figure

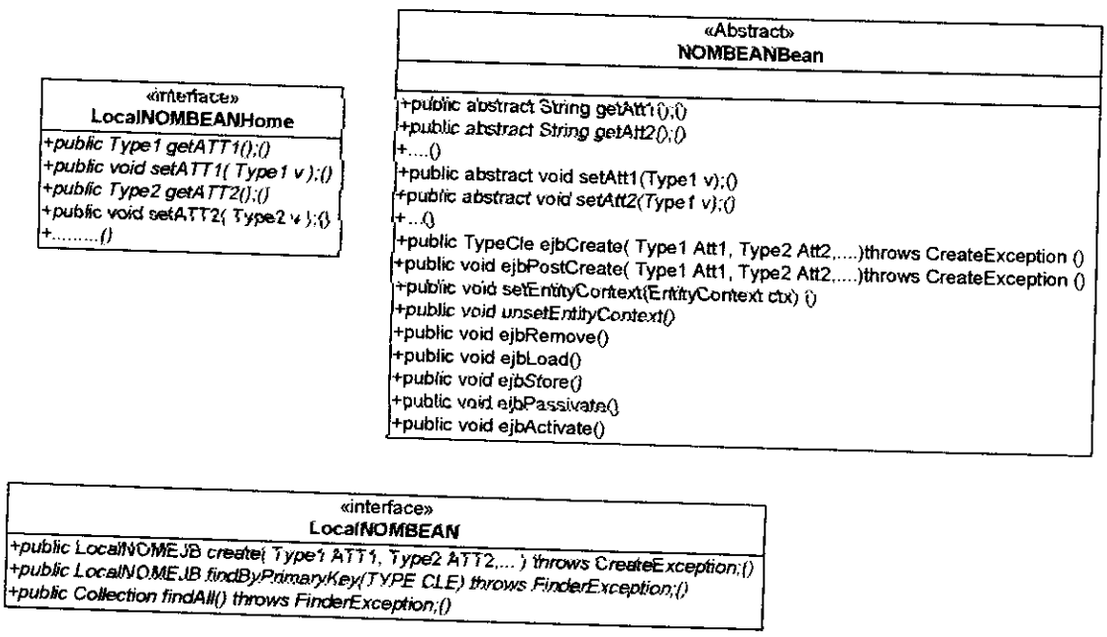


Figure 6.9: La conception interne du EJB Entity cle1

Nous appliquerons ce modèle sur les EJB :
TYPE_RACORDEMENT, MODEME, DEBIT, SERVICE_COMP, REDUCTION_ACC,
TARIF_VOLUME, DEGRESSIVITE_VOL,
DATE_VALEUR, REDUCTION_INT, REDUCTION_NAT
GROUPE_NAT, PAYS, TYPE_JOUR, TRANCHE, STATION, COMMUNICATION
Donc 16 *3 fichiers à créer.

➢ La clé est la composition des clés étrangère:

Nous supposons que le EJB est appelé **NOMEJB** et que la clé est composée de :

CLE1 de type **TYPE1** qui référence le EJB **EJB1**

CLE2 *de type* ***TYPE2*** *qui référence le EJB* **EJB2**

Les ELB qui ont une clé primaire composée doivent définir une classe qui décrit cette clé.

La conception interne de ce type d'EJB est représentée dans la figure

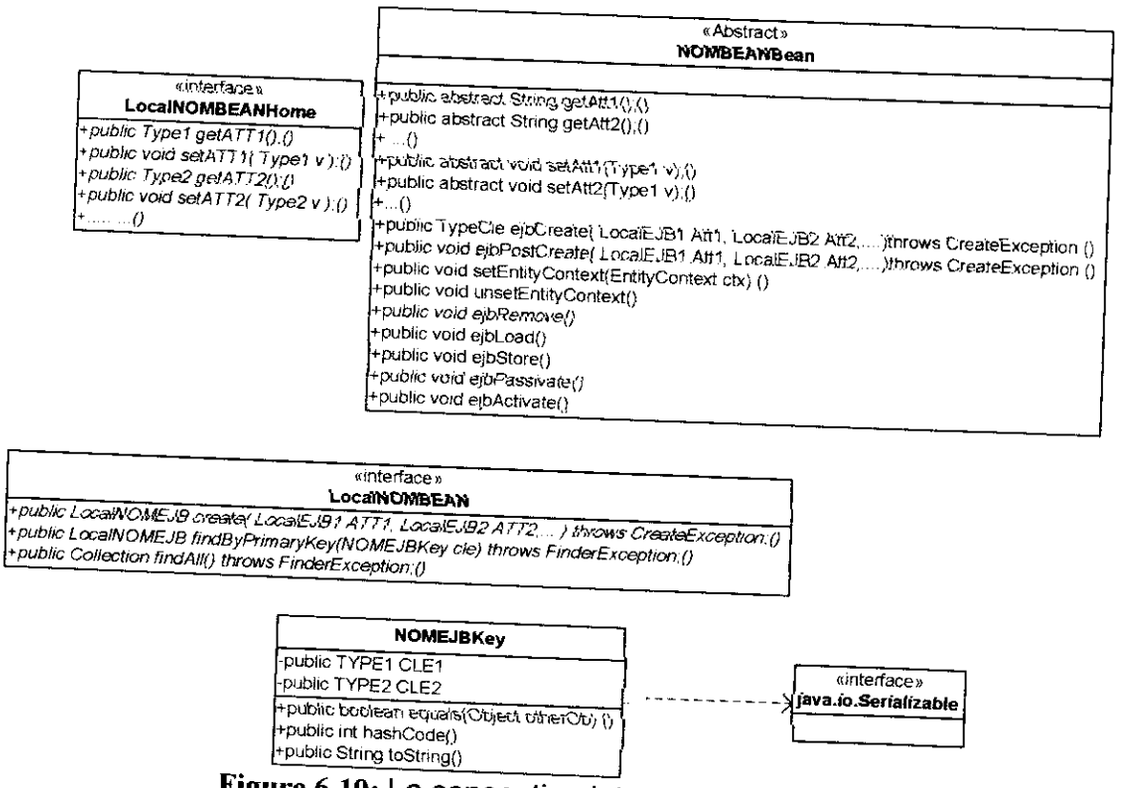


Figure 6.10: La conception interne du EJB Entity cle2

Nous appliquerons ce modèle sur les EJB : `ABON_TYPE_RACOR, ABON_DEBIT, ABON_MODEM, ABON_SERVICE_COMPL, AFFECTER_TARIF_MINNAT, AFFECTER_TARIF_MININT, AFFECTER_TARIF_VOLNAT, AFFECTER_TARIF_VOLINT, REDUCTION_GROUPENAT, REDUCTION_GROUPEINT, TARIF_MES, TARIF_REDEVENCE.`

Donc (12 * 4) fichiers à générer.

➢ Le EJB comporte un attribut qui est une clé étrangère :

Nous supposons que le EJB est appelé **NOMEJB** et qu'il a un attribut autre que la clé :

ATT1 de type **ATT1** qui référence le EJB **EJB1**

La conception interne de ce type d'EJB est représentée dans la figure

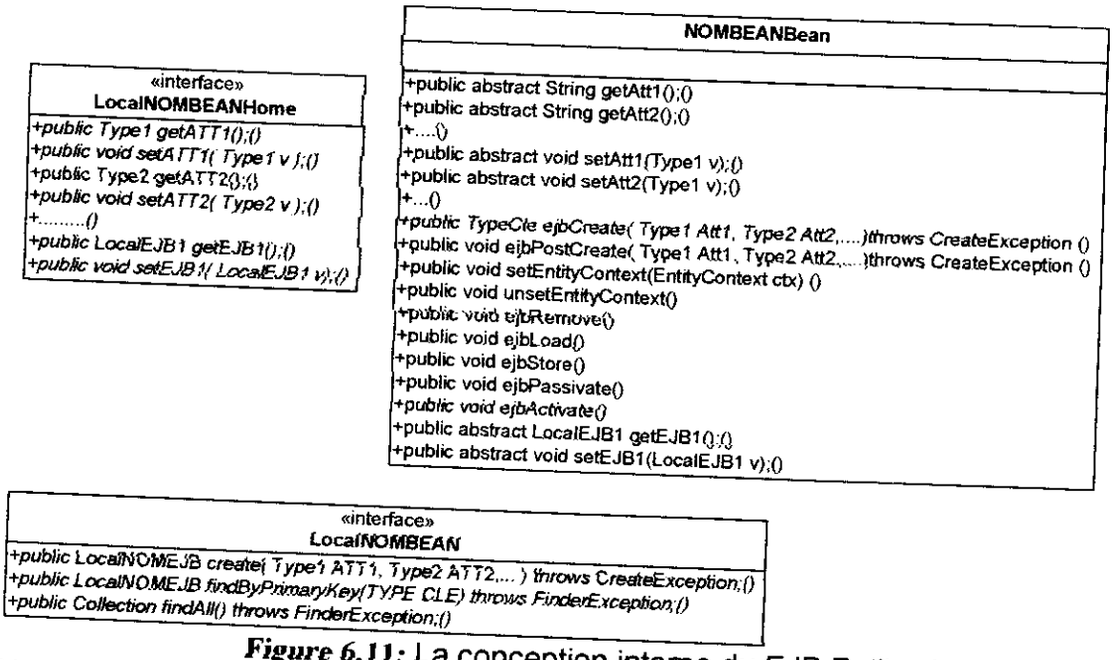


Figure 6.11: *La conception interne du EJB Entity cle3*

Nous appliquerons ce modèle sur les EJB : CONSTRUCTEUR, ABONNEMENT, CLIENT, JOUR_SP, GROUPE_INT.

Donc (5*3) fichier à créer.

2) Le module session :

i. Conception interne :

Le module de session comporte un EJB de type session que nous l'appelons **FaçadeEJB.**

La conception interne de **FaçadeEJB** est représentée dans la figure :

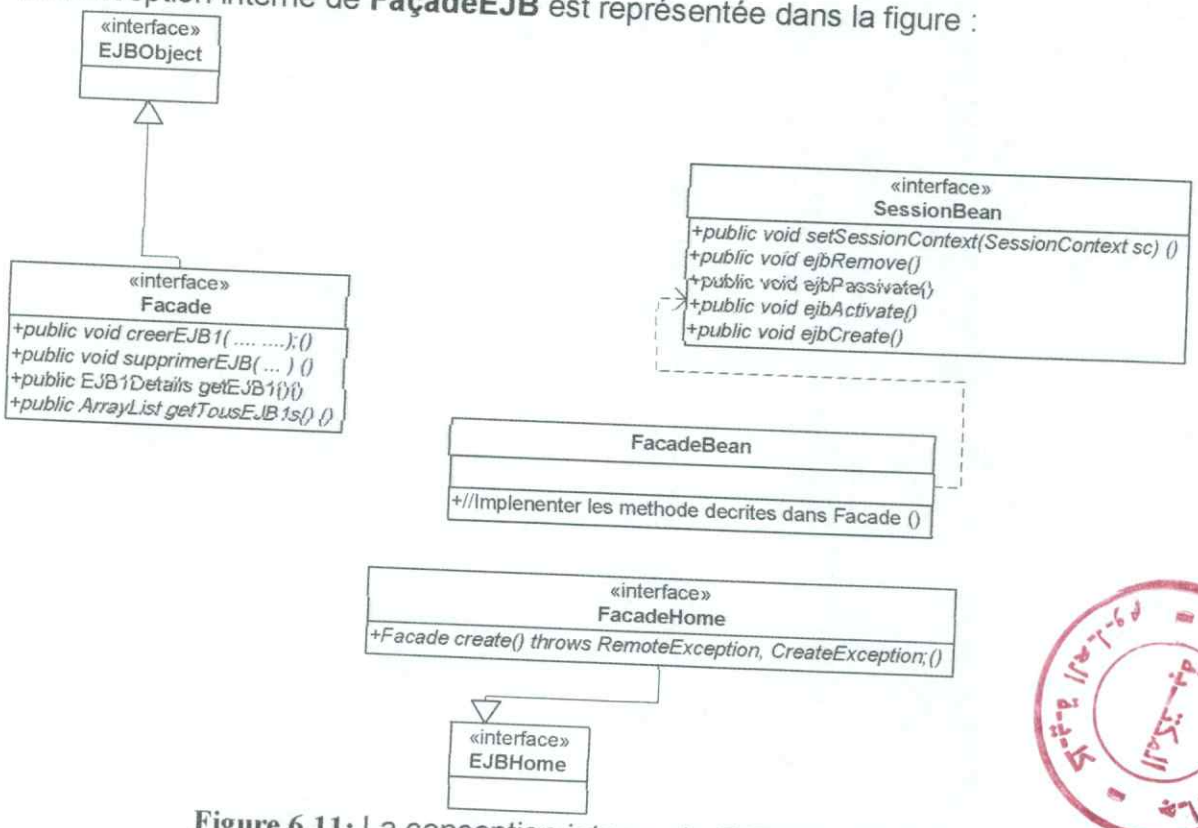


Figure 6.11: La conception interne du EJB Session Façade

➢ *Générer les EJB(entity et session):*

Nous remarquons que nous écrivons beaucoup de code pour créer un EJB et pour définir ses attributs.

Nous constatons la même chose pour **FaçadeEJB**

Le fait que nous écrivons la même chose pour chaque EJB, la création d'un générateur du code source EJB est réalisable.

Les entrées de ce générateur sont :

Nom de l'EJB.

Nom de l'Attribut [PK] [PFK]

PK: clé primaire.

PFK: clé étrangère.

3) Le module Web

Le module web est construit de pages JSP.
Pour chaque EJB nous définissons les pages JSP suivantes :

- Page de création
- Page de consultation
- Page de suppression
- Page de mise à jour.

Donc on aura **34 * 4** page JSP à créer.

- Page index.jsp
- Page login.jsp
- Page loginError.jsp

CHAPITRE VI :
Implémentation

I. Créer la base de données, le pool et la source de données :

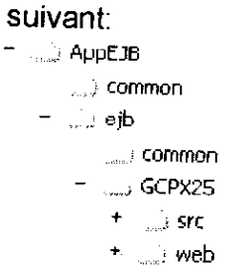
a. La base de donnes :
   a) *Lancer **startconsole.bat***
   b) Taper dans l'url : **jdbc:pointbase:GCPX25BDD**
   c) Taper le user=admin et le password=admin
   d) Choisir *create new database*
   e) Valider, et fermer la console

b. Le pool :
   a) Lancer le serveur d'application : **asadmin start-domain domain1**
   b) Console d'administrateur : lancer le navigateur est taper : http://localhost:4848/asadmin/index.html.
   c) Entrer votre user et password administrateur
   d) Choisir ressource >JDBC > Connection Pool > new
   e) Taper GCPX25Pool dans name.
   f) Sélectionner javax.sql.XADataSource pour le type de ressource
   g) Sélectionner PointBase pour Database Vendor
   h) Cliquer sur Next
   i) Rempliser :
      1. Password=admin
      2. User=admin
      3. DatabaseName= « jdbc:pointbase:server://localhost:9092/GCPX25BDD»
   j) Cliquer sur save.
   k) Lancer le serveur de base de données(Start PointBase).
   l) Pour tester si le pool est correcte cliquer sur **ping.** Le message **Ping succeeded** doit apparaître.

c. La source de données :
   a) ressource >JDBC > JDBC ressources > new
   b) Rempliser :
      1. **jdbc/GCPX25** dans JNDI name.
      2. *GCPX25Pool* dans Pool Name
   c) Valider.

2- Compilation :

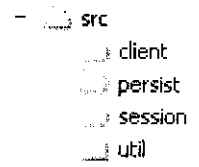
a. Créer l'arborescence suivant:



b. Copier les fichiers : **build.properties, targets.xml** du répertoire j2eetutorial14\exemples\common vers AppEJB\common

c. Copier les fichiers : **build.properties, targets.xml** du répertoire j2eetutorial14\examples\ejb\common vers AppEJB\ejb\common
d. Copier les fichier : **build.xml** du répertoire j2eetutorial14\examples\ejb\cmporder vers AppEJB\ejb\GCPX25.

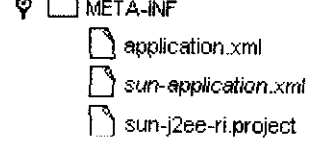
e. Créer l'arborescence suivante dans le dossier src :



f. *Implémenter les différents EJB et les maîtres dans leur dossier approprier* : Les EJB persistant dans le dossier persist, les EJB session dans le dossier session, les utilitaires dans le dossier util, et l'application de teste dans le dossier client.
g. Compilation :
    a) Aller a :AppEJB\ejb\GCPX25
    b) Exécuter la commande : **asant build** : cette commande compile tous les fichiers du dossier src et créer un dossier build qui contient les **.class**

3- Créer l'application :

a. Lancer Deploytool
b. Choisir file > new > Application
c. *Rempliser* :
    a) AppEJB\ejb\GCPX25\GCPX25App.ear dans application file name
    b) GCPX25App dans Application display Name.
    c) Après la sauvgarde le Deploytool crée l'archive : AppEJB\ejb\GCPX25**GCPX25App.ear** qui contient :

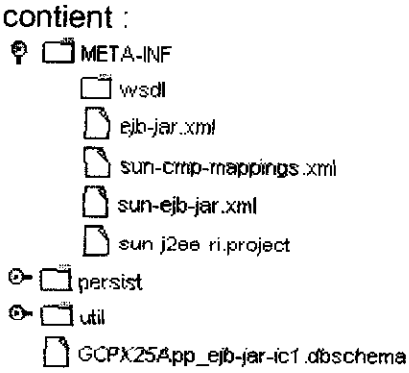


4- Créer le module EJB : persist

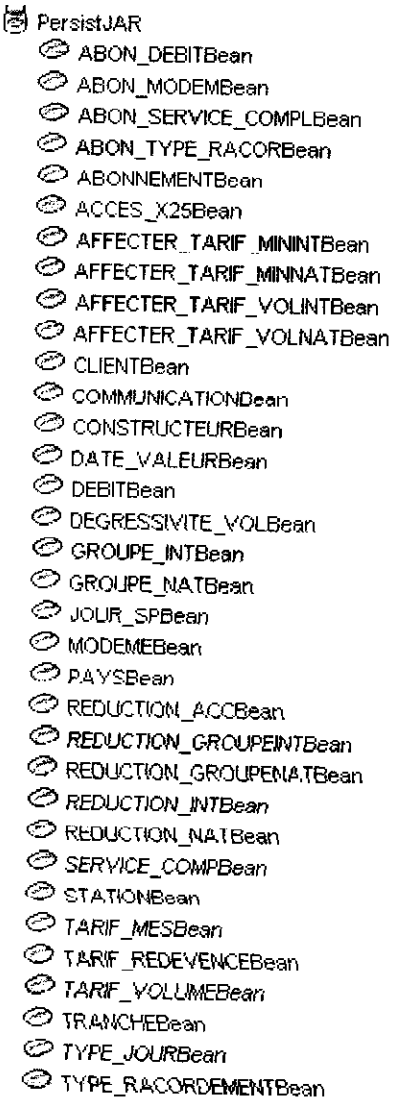
a. Sélectionner file > new > Enterprise Bean >next
b. Selectioner: create new jar file in Application: GCPX25App
c. Taper: PersistJAR dans jar display name.
d. Cliquer sur edit Contents, sélectionner build/persist , build/util et cliquer sur add , et valider
e. Cliquer sur next.
f. Sélectionner Un EJB (persist.DEBITBean) dans entreprise bean class
g. Taper le nom de bean (DEBITBean) dans entreprise bean name
h. Sélectionner (persist.LocalDEBITHome) dans LocalHomeInterface.
i. Sélectionner (persist.LocalDEBIT) dans LocalHomeInterface

j. Cliquer sur next.
k. Sélectionner les attributs qui doivent être persistant.
l. Sélectionner la clé primaire :
    a) Sélectionner un attribut si la clé est un des attributs persistant.
    b) Sélectionner une classe(ex : AFFECTER_TARIF_MININTKey.java) si la clé est definie par une classe.
    c) Sélectionner inconnu primary key si vous voulez que le conteneur créer la clé.
m. Valider.
n. Selectionner file> new > Enterprise Bean >next
o. Sélectionner PersistJAR dans add to existing jar, next
p. Aller a l'étape **f** pour chaque EJB persistant (il y a 35 EJB persistant).
q. Enregistrer.
r. Sélectionner un EJB de l'arbre PersistJAR
s. Sélectionner le menu Entity.
t. Cliquer sur Find/Select Queries.
u. Ecrire le contenu des requêtes. Chaque EJB entity definie au mois la requete findAll().[**select object (p) from DEBIT p**], valider.
v. Aller a **r** pour chaque EJB persistant.
w. Enregistrer.
x. Sélectionner PersistJAR.
y. Sélectionner le menu relationships,
z. cliquer sur add.
aa. Sélectionner le typer de la relation (ex : Many to one)
bb. Sélectionner les deux beans qui participent a la relation.
cc. Accepter ou non la suppression d'un bean si l'autre est supprimé.
dd. Enregistrer.
ee. Sélectionner PersistJAR
ff. Sélectionner le menu General.
gg. Cliquer sur Sun-spesifique Setting
hh. Selectionner CMP database dans view
ii. Taper jdbc/GCPX25 dans JNDI name.
jj. Cliquer sur create database mappings.
kk. Sélectionner Automatically Generate Necessary Tables pour que l'outil gêner les tables qui correspondent a chaque EJB.
ll. Valider et enregistrer :

L'archive PersisteJAR contient :



Les EJB persistant dans l'archive sont :



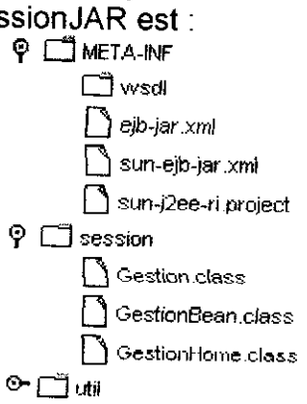
5- Créer le module EJB : Session

a. Sélectionner file > new > Enterprise Bean >next
b. Selectioner: create new jar file in Application: GCPX25App
c. Taper: SessionJAR dans jar display name.
d. Cliquer sur édit Contents, sélectionner build/session , build/util et cliquer sur add , et valider
e. Cliquer sur next.
f. Sélectionner session.GestionBean dans Entreprise java bean.
g. Taper GestionBean dans Entreprise bean name.
h. Sélectionner Statefull Session dans entrprise bean type.
i. Sélectionner session.GestionHome dans Remote Home interface.
j. Sélectionner session.Gestion dans Home interface.
k. Valider et enregistrer.

l. Selectionner le EJB GestionBean. Selectionner le menu EJB Ref's
m. Cliquer sur add

n. Rempliser :
   a) Code Name= ejb/Simple[nom EJB Persistant]
   b) EJB type= Entity
   c) Interface= Local
   d) Home Interface= persist.Local[nom EJB Persistant]Home
   e) Local/remote interface=persist. [nom EJB Persistant].
   f) Entreprse ean name=ejb-jar-ic1.jar#[nom EJB Persistant]Bean
o. Valider
p. Aller a **m** pour chaque EJB Entity.

Le contenu de l'archive sessionJAR est :



L'archive sessionJAR a :

6- Créer le module Web

a. Sélectionner file > new > web component >next
b. Selectioner: create new jar file in Application: GCPX25App
c. Taper: GCPX25WAR dans jar display name.
d. Cliquer sur édit Contents, sélectionner tous les fichier et les dossiers du répertoire **web**, et valider
e. Cliquer sur next. Selectionner servlet.next
f. Selectioner le fichier index.jsp
g. Cliquer sur next > finish.

Le contenu de l'archive AdminWar est :

7- Créer le module Web : ClientWar : idem que AdminWar.
8- Sélectionner AdminWar et cliquer sur EJB ref's
9- Cliquer sur add :
10- Remplir :
   a. Coded Name= **ejb/SimpelGCPX25Session**.
   b. EJB Type= Session.
   c. Interface= Remote.
   d. Home Interface= session.GestionHome
   e. Local/Remote interface= session.Gestion
   f. JNDI name= GestionBean

   Valider.
11-Affectation des droits d'accès.
   a. Créer les Utilisateurs et les groupes :
      a) Lancer la console d'administrateur.
      b) Sélectionner configuration > Security > realms > file
      c) Cliquer sur manage user > new
      d) Rempliser :
         1. User Id.
         2. password.
         3. confirm passord.
         4. groupe.
      e) valider
      f) pour créer d'autre utilisateur aller a c)
   b. Creation des roles :
      a) Sélectionner GCPX25App > rôle >add
      b) Taper : tous
      c) Add
      d) Taper clientèle
   c. Affectation des rôles aux utilisateurs :
      a) Sélectionner AdminWar > security

b) Sélectionner la méthode : Form based comme méthode d'authentification.
c) Cliquer sur Setting .et Rempliser :
   1. Realms name= file
   2. Login Page= /logon.jsp
   3. Login erreur=/logonError.jsp
d) Cliquer sur add contraint. Selectionner SecurityContrainte
e) Cliquer sur add collection.
f) Cliquer sur edit collection. Cliquer add URL Pattern
g) Taper : /* et valider.
h) Sélectionner GCPX25App> sun-specific-setting
i) Choisir User to role mappings.
j) Affecter pour chaque role ces utilisateur et ces groupes
k) Valider et sauvgarder.

12- Déployer l'application :

a. Vérifier que le serveur d'application et le serveur de base de données sont lancés
b. Aller a Tools > deploy et valider.

13- Lancer l'application :

a. Lancer le navigateur et taper : **http://localhost:8080/GCPX25**

CONCLUSION GENERALE

CONCLUSION GENERALE :

Vis-à-vis d'Entreprise Java Bean :

Nous avons constaté que les EJB permettent de se concentrer sur le code métier des composants et de laisser le conteneur s'occuper des services comme la persistance, la sécurité et les transactions... etc.

Malgré l'avantage décrit dans le point précèdent la maintenance des objets persistants reste difficile. Cela a été confirmé par SOPRA qui a constaté qu 80% des développeurs J2EE n'utilisent que la partie servlet/JSP

Vis-à-vis de l'application :

Maintenant que les communications sont sauvegardées, et que l'interface utilisateur est réalisée, nous envisageons de créer les EJB et les JSP nécessaires pour la facturation des clients.

Vis-à-vis des outils développés:

- Nous envisagerons d'améliorer notre générateur de code en ajoutant la notion d'héritage dans le module persistant.
- Nous envisagerons de créer un nouveau générateur qui produit les descripteurs XML pour les EJB générés par le générateur de code.

REFERENCES

[CCC03] Cyril Carrez "**Contrats Comportementaux pour Composants**". Thèse de doctorat de l'École Nationale Supérieure des Télécommunications.2003

[CRM]	Pascal Nicolas. "**Cours de réseaux Maîtrise**"
[EJB]	Lionel Seinturier. "**Enterprise Java Beans**". Université Pierre & Marie Curie
[LIJ00]	Larne Pekowsky. "**Lintro JSP**" .2000
[LRI]	*Dominique LALOT.* " **Les Réseaux Informatiques**"
[PTJ]	F. Martini . "**Présentation des JSP Tag Libraries (Taglibs)** "
[SAA]	Frédéric BON Clever Age. "**Les serveurs d'applications dans les architectures n-tiers**"
[TJT04]	E Armstrong ,J Ball, S Bodoff, D Carson, I Evans, D Green, K *Haase, E Jendrock.* " ***The J2EE™ 1.4 Tutorial*** " *December* 2004
[VAN99]	Rémi LEBLOND. " **Vers une architecture n-tiers** ".**1999**
[01] [02] [03]	www.developpez.con www.sun.com **www.infin-fr.com**